**University of Alberta**

# Eigenspace Projection Clustering Method for Inexact Matching of Graphs and Generic Relational Structures

by

**Serhiy Kosinov**

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2002

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-81423-8

Canada

**University of Alberta**

**Library Release Form**

**Name of Author**: Serhiy Kosinov

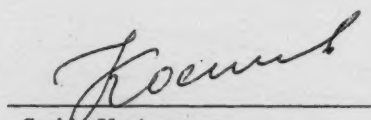
**Title of Thesis**: Eigenspace Projection Clustering Method for Inexact Matching of Graphs and Generic Relational Structures

**Degree**: Master of Science

**Year this Degree Granted**: 2002

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.



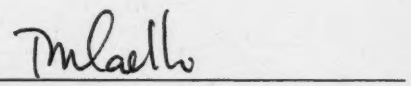
Serhiy Kosinov
8913, 112th St., apt. 3A(D)
Edmonton, AB
Canada, T6G 2C5

**Date:** June 13, 2002

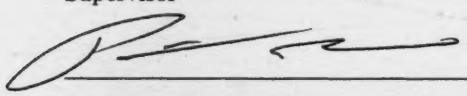
**University of Alberta**

**Faculty of Graduate Studies and Research**

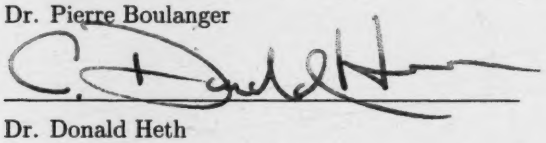
The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Eigenspace Projection Clustering Method for Inexact Matching of Graphs and Generic Relational Structures** submitted by Serhiy Kosinov in partial fulfillment of the requirements for the degree of **Master of Science**.



Dr. Terry Caelli
Supervisor



Dr. Pierre Boulanger



Dr. Donald Heth
External Examiner

**Date:** 13/06/02

# Abstract

In this thesis we show how the inexact graph matching problem can be solved using methods based on the projections of vertices (and their connections) into the eigenspaces of graphs - and associated clustering methods. Our analysis points to deficiencies of recent eigenspectra methods though demonstrates just how powerful full eigenspace methods can be for providing filters for such computationally intense problems.

The proposed eigenspace projection clustering (EPC) method for graph matching relies on the adjacency matrix representation of graphs and deploys computationally efficient mathematical procedure of eigendecomposition, which allows it to be practically useful for large scale problems. An important feature of the EPC method is its ability to match graphs with substantially different number of vertices, which obtains as a result of using renormalization techniques. Also, a new correspondence clustering algorithm designed to be an integral part of the matching procedure enables the EPC method to discover a richer set of correspondence relationships, such as associations among structurally similar groups of vertices and subgraphs, in addition to the one-to-one type of vertex correspondence.

The formulation of the EPC method is remarkably general, which allows it to be adapted for a variety of practical applications that involve matching graphs and generic relational structures. In the presented thesis, this property of the method is demonstrated by applying it to two different problem settings that originate from the domains of text (information retrieval) and image (shape matching) processing. Encouraging results achieved in both of the applications confirm the properties of the EPC method of being general, easily adaptible and practically useful.

To my parents

# Acknowledgements

I would like to gratefully acknowledge the enthusiastic supervision of Dr. Terry Caelli during this work. I also thank Dr. Dekang Lin for the valuable help and technical advice regarding the use of his software (*Minipar*) for natural language text parsing, as well as acknowledge my appreciation of the technical assistance, comments and suggestions received from all of the members of AMMI laboratory.

Also, I am grateful to Dr. Sven Dickinson and his colleague Diego Macrini for providing the shock graph shape database for testing purposes and helping me with the needed image processing software. In addition to that, I want to thank the Department of Computer Science for their support for me over the period of my studies.

Of course, this list would be utterly incomplete without me expressing my sincere gratitude to my parents who have been providing me with vitally important emotional support during my studies thousands miles away from home.

# Contents


**1 Introduction** 1
- 1.1 Graph-based methods for exact and inexact matching of relational structures 1
  - 1.1.1 Exact methods . . . 2
  - 1.1.2 Inexact methods . . . 3
- 1.2 Feature- and clustering-based correspondence recovery approaches . . . 6
- 1.3 The proposed method: integrating eigendecomposition, projection and clustering techniques into a single algorithm . . . 8

**2 EPC: Eigenspace projection clustering** 10
- 2.1 Eigenspectra and eigenvectors of graphs . . . 10
- 2.2 Projections and Normalizations . . . 12
- 2.3 Clustering of projections . . . 16

**3 Application one: matching of 2D shapes represented by shock graphs** 23
- 3.1 Shock graph representation . . . 23
- 3.2 Experimental results . . . 24

**4 Application two: information retrieval with natural language processing enhancements** 30
- 4.1 General information retrieval problem setting . . . 30
- 4.2 Enhancing information retrieval with syntactic relevance data . . . 33
- 4.3 Experimental results . . . 39

**5 Conclusions and future work** 43

**Bibliography** 46

**A Aggregate relational lattice construction example** 49

# List of Figures


2.1 Examples of non-isomorphic graphs with identical eigenspectra . . . . . 11
2.2 Example 1: graphs and their projections . . . . . 14
2.3 Clustering of graph eigenspace projections located in separate hyperplanes. The shown distances A and B provide an example of intra-graph and inter-graph distances, respectively. . . . . 18
2.4 Example 2: Clustering of vertex projections of sample graphs Z and T. . . . 21

3.1 Four schematic shape regions and their corresponding shock types (adapted from [34]) . . . . . 24
3.2 A sample shape of a mallet and its shock graph representation . . . . . 25
3.3 Shape similarity summary table as computed according to Equation 2.6a (a box drawn around a similarity value corresponding to best matching pair) . 28
3.4 An example of comparison of shapes within the same category (the calculated similarity values are shown above each shape instance) . . . . . 28
3.5 Two sample shapes BOXER-10 and BOXER-24, and their shock graph representations . . . . . 29

4.1 Graph representation of syntactic relations among the words in sample queries $q_1$ and $q_2$ . . . . . 35
4.2 Eigenspace projections of the parse trees of queries $q_1$ and $q_2$ (the following lines are drawn in order to visualize the distances between projections of the matching index terms: a solid line for keyword *dynamic*, a dashed line for keyword *operating*, and a dotted line for keyword *system*) . . . . . 37
4.3 Distances between the eigenspace projections of matching keywords from the two sample queries $q_1$ and $q_2$ . . . . . 38
4.4 Parse trees of sample sentences from document 27 and query 13 . . . . . 40
4.5 Eigenspace projection of syntactic structure of query 13 and a sample sentence from document 27 (ADI collection) . . . . . 41
4.6 Eigenspace projection of syntactic structure of query 13 and aggregated syntactic relational lattice of document 27 from ADI collection (query keywords are depicted as red diamonds, while document terms are shown as blue circles; connecting lines are drawn in order to visualize the distances between projections of the matching index terms: *criteria*, *evaluation*, *retrieval*) . . . 42
4.7 Performance of baseline (B.) and modified (M.) retrieval systems for the subset of short queries from ADI and CISI text collections . . . . . 42

# List of Tables


1.1 Summary of important properties of various methods graph matching . . . . 5

2.1 Cluster decomposition obtained from correspondence clustering of the two graphs $Z$ and $T$ . . . . . 21

3.1 Cluster decomposition obtained from correspondence clustering of the two shock graphs for shapes BOXER-10 and BOXER-24. Each cluster enumerates the vertices from the two graphs that are grouped together and lists the part of the body that the vertex approximately comes from, in brackets. . . . . 27

4.1 A mini-set of documents consisting of several book titles . . . . . 31
4.2 Vector representation of documents from the book title dataset (each document is a column vector) . . . . . 32
4.3 Similarity measures and the resulting ranking for a sample query and documents from the book title dataset . . . . . 32
4.4 Parameters of the two text collections used in experiments . . . . . 39
4.5 Performance of baseline (B.) and modified (M.) retrieval systems for the subset of short queries from ADI and CISI text collections . . . . . 41

# Chapter 1

# Introduction

There exist numerous practical application areas in which matching of complex relational structures, i.e. the entities that consist of a number of inter-related components joined together in a certain way, plays a key role. For instance, in chemistry, one such area is an analysis of chemical structures[27], in machine learning, it is a study of common substructures of different concepts[2], in computer vision and pattern recognition, matching and analysis of relational structures are the core problems in such tasks as character recognition, object identification, shape analysis[43, 15]. In all of these and many other areas various methods have been developed for solving such problems. In this thesis we present a method for matching graphs and generic relational structures, together with some of its applications, that combines the advantages of eigendecomposition, clustering and projection techniques in an attempt to overcome several important shortcomings of the previously developed solutions to be discussed in the following sections.

## 1.1 Graph-based methods for exact and inexact matching of relational structures

In the majority of the above mentioned application areas the problem of complex relational structure matching is approached with the aid of well-established graph theory formalisms, which provide an abstract and very powerful representational framework.

Two concepts from graph theory that are of fundamental importance from the point of view of the analysis and matching of relational structures are those of *graph isomorphisms* and *subgraph isomorphisms*. When matching two graphs $G_1$ and $G_2$ by means of graph isomorphism one looks for a bijective mapping between vertices of $G_1$ and $G_2$ such that the

structure of the edges is preserved by the sought mapping. When one of the graphs involved in matching is larger than the other, e.g. graph $G_2$ contains more vertices than $G_1$, then the matching is performed by a ***subgraph isomorphism***, which involves finding a subgraph $S$ of graph $G_2$ such that graphs $G_1$ and $S$ are isomorphic.

Practically, an implementaion of a general exact graph matching algorithm based on the notion of subgraph isomorphism leads to solving an NP-complete problem[12]. Consequently, most research efforts in this area have been directed at improving performance of the matching algorithms, both in terms of memory requirements and algorithms.

### 1.1.1 Exact methods

One of the earlier advances in the field of exact graph matching is due to Ullman's procedure [37] for graph and subgraph isomorphisms. The procedure is based on a standard combinatorial search algorithm, such as one documented in [6], that operates as follows. For a pair of graphs $G_1$ and $G_2$, the vertices of $G_1$ are mapped one by one onto the vertices of $G_2$. At the same time, the algorithm has to check the obtained mapping at every iteration in order to ensure that the edge structure is preserved by the mapping process, and backtrack when necessary. The enhancement introduced by Ullman combines backtracking together with a forward-checking procedure that allows to reduce the number of backtracking steps, and thus the whole search space, significantly.

Among the more recent additions are, notably, the Nauty and VF[5] algorithms that speed up the search process even more. The former algorithm, designed by McKay[21], is based on trasformations that reduce the graphs being matched to a canonical form on which the testing for the isomorphism is substantially faster, while the latter algorithm adopts a depth-first strategy and uses a set of isomorphism-specific rules to prune the search tree. There have also been developed methods that use decision trees[23], association graphs and maximal clique finding techniques[26] for solving graph and subgraph isomorphism problems. However, even with a significant improvement in performance, isomorphism-based methods for exact graph matching may not be used in many practical tasks because of that very property of being exact.

In most real-world applications we deal with somewhat noisy data, which typically introduces distortions into graphical representations, and hence the need to have matching

methods that can handle such distortions gracefully. For this purpose, there exists a substantial body of research that aims to overcome the problem of noise, distortion and error in graphical representation of relational structures by incorporating and relying on explicitly modeled errors that are encountered during the process of matching. These approaches are generally categorized as *inexact matching* techniques that provide a means of performing error-tolerant, or error-correcting, graph matching.

### 1.1.2 Inexact methods

In the domain of inexact graph matching there also exist several groups of methods that differ from each other in the way that errors are modeled and the problem of matching itself is posed. A rather large family of methods capable of inexact graph matching is represented by a set of approximate, or, continuous optimization algorithms. One of the essential advantages of the approximate algorithms is that they, in contrast to the optimal algorithms discussed above, no longer require exponential time to solve the matching problem, but can cope with the task in polynomial time. However, this property is not without cost: the approximate algorithms may converge to local optima and thus are not always guaranteed to find the correct solution[22].

**Continuous optimization methods**. Nevertheless, there have been a number of contributions that demonstrate the value of the continuous optimization techniques with respect to the problem of inexact graph and subgraph matching. In [7], the authors utilize Genetic Algorithms (GA) to optimize the vertex to vertex mappings encoded as sequences of chromosomes using the standard means provided by the GA framework, such as recombination (or, crossover), mutation, and fitness function estimation (or, selection). Myers and Hancock [25] make a step further and develop the actual GA operators of recombination, mutation and selection that work on graphs themselves rather than on the encoded vertex mappings. Feng et al.[11] propose to solve the inexact graph matching problem using Neural Networks by representing each vertex to vertex mapping by a neuron in a Hopfield network whose output is subsequently optimized. Kittler et al.[14] describe a method for inexact graph matching based on probabilistic relaxation, in which the individual probabilities of each vertex to vertex mapping are corrected until the probability for the whole set of vertex to vertex mappings is maximized. Another approach that also situates the problem of inexact

graph matching in the probabilistic framework is due to Luo and Hancock [17]. The authors adopt an explicit model of the correspondence errors encountered during graph matching with the aid of the Bernoulli probability distribution, and hence are able to devise a graph matching likelihood function that allows one to estimate the conditional likelihood of one graph given the other and determine the best possible vertex to vertex mapping by using Expectation Maximization (EM) and Singular Value Decomposition (SVD) techniques.

**Pattern recognition methods**. Another important direction of research into inexact graph matching emerged from the field of structural pattern recognition and is mainly focused on developing a well-defined measure of structural similarity. In this area, Eshera and Fu[9], Sanfeliu and Fu[28], as well as Bunke[3] have studied the matching problem using *graph edit distance*, a concept that provides a measure of dissimilarity of two given entities and has its origins in the domain of strings. The general idea of such an approach is to compare a pair of graphs by finding a sequence of edit operations, such as edge/vertex deletion, insertion or substitution, that transforms one graph into the other, whereas the dissimilarity, or distance, of the two graphs is said to be the minimum possible cost of such a transformation. Furthermore, Bunke[13] extends this work by developing other important notions, such as the weighted mean and generalized median of a pair of graphs, which makes it possible to apply clustering and self-organizing map techniques in the domain of graphs. Similar to these contributions is the effort of Tirthapura et al.[36] in using the classical Levenshtein distance for shock graph matching.

**Spectral methods**. There also exists a family of graph matching techniques, generally known as spectral methods, that seek to represent and distinguish structural properties of graphs using eigenvalues and eigenvectors of graph adjacency matrices. The most valuable characteristics of such methods include being invariant to edge/vertex reordering, ability to map a graph's structural information into lower-dimensional spaces and stability under minor perturbations. On top of that, the eigendecomposition technique itself is far less computationally expensive as compared to the advanced combinatorial search procedures. Among recent developments in this field is the Umeyama's[38] formulation for same-size graph matching. The method aims to obtain a matrix $P$ that would minimize the structural difference between two graphs $G_1$ and $G_2$ defined as: $J(P) = ||PA_{G_1}P^T - A_{G_2}||$, where $A_{G_1}$, $A_{G_2}$ are the adjacency matrix representations of the two graphs being compared. The author

proves that the sought permutation matrix $P$ can be derived from the eigendecomposition of the graphs' adjacency matrices, which can be used as an efficient and computatinally robust technique for matching of graphs with the same number of vertices.

Dickinson et al.[32] considered indexing hierarchical structures, or directed acyclic graphs (DAG), with topological signature vectors being obtained from the sums of adjacency matrix eigenvalues. The method is especially suitable for image and shape matching applications because of its valuable property of being able to deal with the problem of occlusion, which is achieved by constructing the topological signature vectors incorporating eigenvalue metrics derived from a broad range of subgraphs of a given DAG.

A brief summary of important properties of the above discussed methods is given in Table 1.1

Table 1.1: Summary of important properties of various methods graph matching

| Method / Procedure | Algorithm | Model | Optimal | Inexact |
|---|---|---|---|---|
| Ullman's procedure | combinatorial search that combines backtracking and forward-checking | graphs | yes | no |
| McKay's "Nauty" algorithm | Reduction of a graph to a canonical form to enhance search process | graphs | yes | no |
| VF, VF-2 algorithms | Use of isomorphism-specific rules to prune search tree | graphs | yes | no |
| Genetic algorithms, Hancock et al. | Use of GA operators of mutation, crossover and selection in search for isomorphic match | sequences of chromosomes | no | yes |
| Neural networks, Feng et al. | Optimization of output of a Hopfield neural net representing vertex to vertex mapping | neural network | no | yes |
| Kittler's method et al. | Probabilistic relaxation process applied to individual probabilities of vertex to vertex mapping | probabilistic | no | yes |
| Approaches of Fu, Eshera, Sanfeliu, Bunke | Use of structural similarity measure based on the concept of graph edit distance | graphs | yes | yes |
| Umeyama's method | Deriving an optimal permutation matrix via eigendecomposition | graph adjacency matrices | no | yes |
| Hierarchical structure indexing method, Dickinson et al. | Indexing graphs with topological signature vectors composed of sums of subgraph eigenvalues | graph and subgraph adjacency matrices | no | yes |

## 1.2 Feature- and clustering-based correspondence recovery approaches

Although the previous list of approaches seems quite extensive and diverse, there also exists a substantial body of research that neither explicitly adopts graph theoretical representation and formalism nor poses the problem as the one of graph matching, but nevertheless can be shown to be important in the analysis and matching of graphs and generic relational structures. Indeed, one may consider the problem of graph matching as that of establishing correspondence between structurally similar components, i.e. vertices, edges and subgraphs. Hence, it may be beneficial to explore various other non-graph-based methods of correspondence recovery with respect to their applicability to the problem of matching graphs.

**Feature-based correspondence**. One such method for solving correspondence recovery problem was developed by Scott and Longuet-Higgins[30]. The authors' approach aims to find corresponding features between two related images with the aid of linear algebra techniques[1]. First, a proximity matrix is formed such that each of its elements $(i, j)$ represents a Gaussian-weighted distance between the $i$-th feature of one image and $j$-th feature of the other image. Then, the obtained proximity matrix is transformed into a product of three matrices using the singular value decomposition technique. Finally, the middle matrix of singular values is modified, so that all of its diagonal elements are set to 1, and substituted back into the decomposition product rendering an "association matrix". After that, the corresponding features of two images can be determined using the following simple rule: $i$-th feature in one image corresponds to $j$-th feature in the other image if the association matrix element in position $(i, j)$ is close to 1, and there is no correspondence otherwise.

The Scott and Longuet-Higgins' approach created the foundation for further work on feature-based correspondence using linear algebra techniques and, quite naturally, served as a starting point for several important contributions in the area, such as the method proposed by Shapiro and Brady[31]. Similarly to the approach taken by Scott and Longuet-Higgins', Shapiro and Brady focused on sets of features of two images used to create Gaussian-weighted proximity matrices. In contrast to the Scott and Longuet-Higgins' approach, however, there are two matrices that the method needs to deal with, each of which records

[1] assuming correspondence between the coordinate systems.

intra-image distances among the features of a given image[2]. Both proximity matrices are then submitted to a linear algebra technique widely used in spectral methods for graph matching, namely, - eigendecomposition. The components of the eigenvectors obtained via the eigendecomposition procedure are subsequently combined to represent two sets of image features with *feature vectors* that, in turn, are correlated to each other yielding the feature association matrix. Having obtained the association matrix, the corresponding features of two images can be determined using the following simple rule: $i$-th feature in one image corresponds to $j$-th feature in the other image if the association matrix element in position $(i, j)$ is close to 0, and there is no correspondence otherwise.

Taking into account the above discussion, one may notice that feature-based correspondence recovery methods have a lot in common with the approaches directed at solving graph matching problems. Indeed, all of the spectral graph matching methods adhere to the same mathematical framework of matrix representation and deploy similar linear algebra techniques, such as eigendecomposition. Also, there is a substantial resemblance between certain graph matching methods and feature-based correspondence recovery techniques that becomes obvious once the equivalence between graph vertices and image features, as well as between graph edges and intra-image proximities, is considered. Therefore, it may be beneficial to try to exploit the apparent computational advantages found in the feature-based correspondence recovery methods for solving the graph matching problem.

**Clustering-based correspondence**. In addition to the above described methods for feature-based correspondence, there are several other techniques that, on the one hand, are even further away from the topic of graph matching, but, on the other, have a number of important ideas that can be regarded quite relevant for the problem in question. Namely, these are clustering-based methods for correspondence detection that originate from the area of text processing[20, 19, 45][3]. According to Marx et al.[19], the operation of a general clustering-based correspondence recovery method may be described as follows. A method takes two separate datasets as inputs, between subsets of which a correspondence relation needs to be established. Then, as in the standard data clustering framework, each of the two datasets is partitioned into disjoint subsets. Unlike the standard clustering case, each

[2]as opposed to inter-image distances, as it was specified in the Scott and Longuet-Higgins' method.

[3]Unfortunately, there is no consistency in the literature as for how these methods should be called. Various sources refer to the same idea as "bipartite clustering", "coupled clustering" or "correspondence clustering"

subset is coupled with a corresponding subset of the other dataset. Each such pair of coupled subsets can be viewed as a unified coupled cluster, containing elements from the two datasets. Subsequently, within any given cluster, the elements from one dataset are said to correspond to those from the other dataset.

From the point of view of the graph matching problem, an apparent advantage of using clustering-based correspondence recovery methods is the richness of correspondence relations that can be found and the notion of proximity for inexactness. That is, if a prototype method for detecting correspondence via clustering, such as the one outlined above, is applied to a pair of datasets whose elements represent vertices of two graphs, then the resulting clusters may help determine the correspondence relationships not only between individual vertices, but also betwen larger vertex groups and, potentially, whole subgraphs. This property could be useful when matching graphs of considerably different sizes, in which case no reasonable one-to-one vertex matching may exist. Hence, it may prove advantageous to borrow some valuable properties inherent to clustering-based correspondence approaches when solving graph matching problems.

## 1.3 The proposed method: integrating eigendecomposition, projection and clustering techniques into a single algorithm

The above sections have presented some of the previously developed methods for graph matching, and covered a number of approaches indirectly related to the problem of inexact subgraph matching. In the following discussion, we will propose a new method designed for the matching of graphs and generic relational structures that strives to combine the advantages and relevant ideas mentioned earlier, while at the same time trying to avoid the shortcomings of the above described solutions. Namely, our method aims to achieve the following goals of being able to:

1. perform inexact graph matching necessary for dealing with approximations, distortions and errors in relational data,

2. take advantage of computationally efficient linear algebra techniques, such as eigendecomposition, in order to make the process of matching both robust and computationally tractable for large scale problems,

3. allow for a richer spectrum of correspondence relations to be discovered, which is necessary for matching graphs with substantially different number of vertices,

4. maintain a sufficient level of flexibility and generality in order for the method to be easily applied in various tasks that incorporate matching of graphs and generic relational structures.

Details follow.

# Chapter 2

# EPC: Eigenspace projection clustering

## 2.1 Eigenspectra and eigenvectors of graphs

As mentioned above, the basic technique deployed in the majority of spectral methods and feature-based correspondence recovery approaches is eigendecomposition. In general, for undirected graphs, it is expressed as follows:

$$A = VDV^T \tag{2.1}$$

where $A$ is the square symmetric adjacency matrix of a graph, whose entry $a_{ij}$ at the place $(i,j)$ is equal to one if there exists an edge that connects vertex $i$ with vertex $j$, and zero otherwise; $V$ is an orthogonal matrix whose columns are normalized eigenvectors of $A$, and $D$ is a diagonal matrix containing the eigenvalues $\lambda_i$ of matrix $A$. The set of the eigenvalues found on the diagonal of matrix $D$ is called the spectrum of $A$, and hence the common name for the family of methods.

One of the most well-known properties of eigendecomposition, and the one that has attracted researchers' attention for the purpose of solving inexact graph matching task in the first place, is that an eigenvalue spectrum of a matrix is invariant with respect to similarity transformations, i.e. for any non-singular matrix $P$, the product matrix $PAP^{-1}$ has the same eigenvalues as $A$. From the view point of the graph matching problem, this means that the derived spectrum of a graph represented by its adjacency matrix is not affected by any arbitrary vertex reorderings, whose influence, or rather lack thereof, is in essence captured by the above vertex permutation matrix $P$. In addition to that, it was proven in [32] that small changes in the structure of a graph, and hence its adjacency

matrix, induce respectively small changes in the magnitudes of the calculated eigenvalues. These two valuable properties of eigenspectrum of, firstly, being invariant with respect to similarity transformations and, secondly, not being overly sensitive to small perturbations in the adjacency matrix data made it look as a very promising candidate for a simple and intuitive solution of inexact graph matching problem.

Still, regardless of however elegant the possible graph matching problem solutions seemed at first in terms of graph eigenspectra, it was proven early on that the spectra of graphs are not unique. An obvious example that dates back to as far as 1957 was discovered by Collatz and Sinogowitz[4], and is shown in Figure 2.1 (the first pair of graphs, 8 vertices).

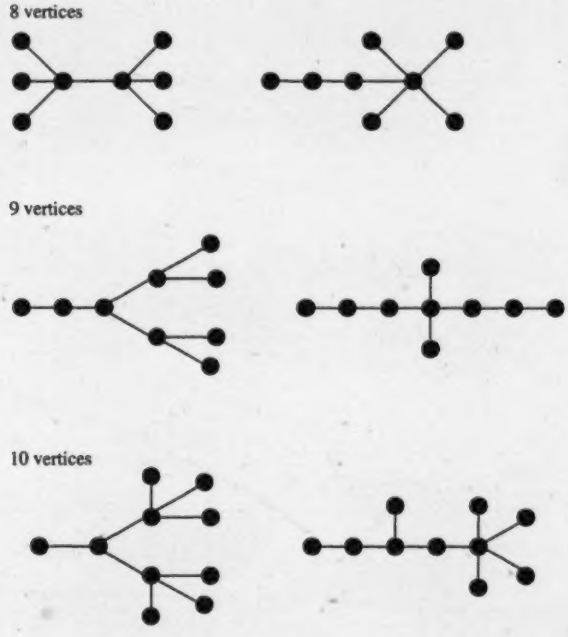



Figure 2.1: Examples of non-isomorphic graphs with identical eigenspectra

The above figure depicts three pairs of non-isomorphic graphs on 8, 9 and 10 vertices, that are nevertheless co-spectral, i.e. the sets of eigenvalues of adjacency matrices of graphs from each pair are identical, and therefore the two graphs cannot be distinguished by relying exclusively on their spectra. Furthermore, Schwenk[29] demonstrated that as the number of vertices gets large, the probability of occurrence of a non-isomorphic co-spectral subgraph pair in any two graphs being compared asymptotically approaches unity. This means that pure spectral methods based solely on eigenvalues are generally not rich enough to fully represent graph structure variability.

Naturally, the above arguments do not add support for spectral methods. However, it is not so difficult to see that this lack of uniqueness can be easily overcome by using graph spectra together with the set of associated eigenvectors, or even by relying on the eigenvectors alone[1] (see Equation 2.1). Another drawback usually attributed to the spectral methods is that they are not extendible to matching graphs of different sizes. For example, the earlier mentioned method developed by Umeyama[38] applies only for graphs of the same size. Nevertheless, these shortcomings can be eliminated by applying projection and normalization operations - the topic of the following section.

## 2.2 Projections and Normalizations

We have seen from the above discussion that the attractive property of eigenspectra of providing a compact representation of graphs' structural information does not come at no cost. There is an accompanying lack of uniqueness that renders a pure eigenspectrum technique unusable for graph matching. But maybe there exist some other methods for deriving a compact, lower-dimensional representation of the relational data expressed by a graph adjacency matrix? For this purpose we might need to have a closer look at the tools available in the field of principal component analysis.

Subspace projection methods, in the principal component analysis (PCA) literature, are conventionally used to reduce the dimensionality of data, while minimizing the information loss due to the decreased number of dimensions. It is performed in the following way. The dataset covariance matrix $\Sigma$ is first decomposed into the familiar eigenvalue and eigenvector matrix product (see Equation 2.1):

$$\Sigma = U\Lambda U^T \tag{2.2}$$

where $U$ is a matrix of eigenvectors ("principal components" of the data), and $\Lambda$ is a diagonal matrix of eigenvalues. The original data is then projected onto a smaller number of the most important (i.e., associated with the largest eigenvalues) principal components as specified in the below equation (and thus, the data's dimensionality is reduced):

$$\hat{x} = U_k^T x \tag{2.3}$$

[1] For instance, recall the previously discussed method for feature-based correspondence recovery developed by Shapiro and Brady[31], where, instead of relying on eigenspectra, the authors use eigenvector components to represent image data by feature vectors.

Here, $\hat{x}$ is the computed projection, $U_k^T$ is the matrix of $k$ principal components in a transposed form, and $x$ is an item from the original data.

Taking the very same approach, we can project vertex connectivity data from a graph adjacency matrix onto a smaller set of its most important eigenvectors. The projection coordinates obtained in this way would then represent the relational properties of individual vertices relative to the others in the lower-dimensional eigenspace of a given graph. In this eigenvector subspace, structurally similar vertices or vertex groups would be located close to each other, which can be utilized for approximate comparison and matching of graphs.

However, in order to be able to use the above projection method for graph matching, it is necessary to resolve the following issues: first, how many dimensions to choose for vertex eigenspace projections? Second, how to ensure the comparability of the derived projections for graphs with a different number of vertices?

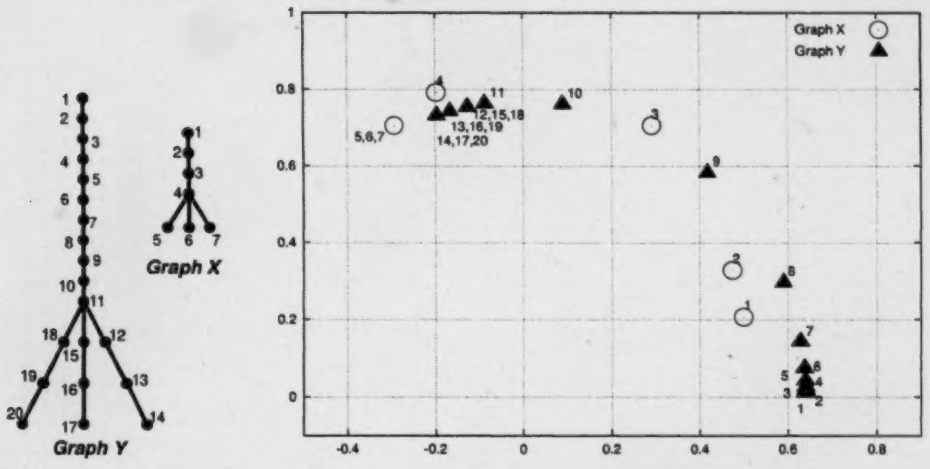
The first question is answered by the relative sizes of the eigenvalues associated with each dimension or eigenvector with non-zero eigenvalue signalling the redundancy of the associated subspaces. That is, for a given pair of graphs one should choose the $k$ most important eigenvectors as the projection components, where $k$ is the smaller value of the ranks of adjacency matrices of the two graphs being compared[2] , as expressed in Equation 2.4.

$$k = min(rank(A_{Graph_1}), rank(A_{Graph_2})) \tag{2.4}$$

This value, however, may be safely decreased, or, more precisely, divided by two, due to the fact that for undirected graphs the adjacency matrices are always symmetric, and so are the opposite components of the derived eigendecomposition. In addition to that, as confirmed by the later practical experiments, one can find an even more precise estimate of a proper value of $k$ by using Bartlett's test for dimensionality[46].

As for the second question, the empirical evidence suggests that an extra step of renormalization of the projections may suffice. Here, the idea is that for the purpose of comparing two arbitrary graphs we need not consider the values of the projections as such, but instead should look at how they are positioned and oriented *relative to each other* in their eigenvector subspace. That is, if projections are themselves viewed as vectors, we disregard their

[2]However, in order to make the two following examples more illustrative, without a loss of generality in the further discussion we will use only 2-dimensional projections, which can be easily depicted in the 2D plane.




(a) Graphs X, Y.

(b) Projections of graphs X and Y into 2D eigenvector subspace.

Figure 2.2: Example 1: graphs and their projections

magnitudes, while only paying attention to their direction and orientation. And this is exactly what projection coordinate renormalization helps us to do: in the end all of the projections are unit-length vectors that can only be distinguished by their orienation, and not by their length. In addition to that, we also carry out a dominant sign correction of the projection coordinates of either of the two graphs being matched so as to align one set of graph vertex projections against the other. The scheme of operation of the sign correction rule is fairly simple. For each eigenvector, the number of its positive and negative components is counted. The eigenvector is then multiplied by $-1$ if the number of negative components is greater than the number of positive components, or left as is otherwise. This caters for the non-uniqueness of signs of eigenvector components and, according to a geometric interpretation, corresponds to setting the direction of the axes in such a way to result in the most compatible alignment between the vertex data using the dominant sign test.

Consider an example with two graphs $X$ and $Y$ depicted in Figure 2.2(a). Although different in size, the two graphs are nevertheless quite similar to each other. In fact, one may see graph $Y$ as an enlarged version of graph $X$. The result of projecting the two graphs into the normalized 2D eigenvector subspace shown in Figure 2.2(b) demonstrates the following two important features of the proposed method: firstly, the projections of

vertices of both graphs follow a similar pattern, which means that it is possible to determine overall structural similarity of graphs with different number of vertices, and secondly, one may also see (by examining the juxtaposition of the projected vertices of both graphs) that graph vertices with similar relational properties tend to get projected into the areas that are close to each other. These properties are quite valuable, and, as such, have the potential to prove useful in solving the graph matching problem. The latter conjecture is confirmed by the experimental results which show that an overall graph similarity can be estimated by comparing the vertex projection distributions with the aid of multi-dimensional extension of Kolmogorov-Smirnov (K-S) statistical test[10]. However, the K-S test becomes a rather computationally expensive procedure if applied to high-dimensional data. Also, it does not help us much to resolve another important issue of the graph matching problem, namely, the one of recovering structurally similar vertex correspondence in a pair of graphs being compared. To this end, we use clustering methods as described in the following section.

However, before considering a detailed description of these clustering procedures, it could be helpful to review the similarities and highlight important distinctions between the proposed graph matching method and that of Shapiro and Brady[31], which, technically, is the closest approach in terms of the underlying mathematical techniques used, in order to have a more precise understanding of the main principles involved in the operation of the proposed method. At this point, the striking resemblance between the Shapiro and Brady's feature-based correspondence approach and the proposed graph matching method can be clearly seen. Indeed, the intra-image proximity matrices (Shapiro and Brady) may be regarded as next of kin for graph adjacency matrices (proposed method), if one observes the similarity between intra-image distances and edges in graphs connecting pairs of vertices. Furthermore, there is a near-equivalence relationship between the mathematical representation of the feature vectors (Shapiro and Brady) and eigenspace vertex projections (proposed method), as shown in Equations 2.5a and 2.5b (using the same notation as in Equation 2.3):

$$\textit{eigendecomposition}: \quad x = UDU^T \tag{2.5a}$$

$$\textit{projections}: \quad \hat{x} = U_k^T x = U_k^T U D U^T \equiv D_k U_k^T. \tag{2.5b}$$

So, as Equation 2.5b says, the projections, similarly to the feature vectors in the approach of Shapiro and Brady, also consist of the components of the eigenvectors derived from the

adjacency matrix eigendecompositions.

Despite all of these common characteristics of the two approaches, there are several distinguishing properties of the proposed method that need to be pointed out.

- First, in contrast to the approach of Shapiro and Brady, the eigenvector components in the vertex projections are scaled by the corresponding eigenvalues (see Equation 2.5b), which means that the more prominent structural qualities of a given graph encoded in the more important "principal components" will be reflected by larger projection magnitudes, and vice versa.

- Second, the proposed method uses an additional stage of projection normalization, i.e. normalization after normalization, or, renormalization[3] , that essentially helps overcome an important obstacle previously encountered in the majority of spectral methods, namely, the inability of the technique to cope with the cases when the number of graph vertices, image features, pixels, and so on, in the two relational structures being matched was substantially different.

- Third, the proposed method uses a more computationally efficient dominant sign correction rule, that for $N$ possible projection axes requires $N$ orientation tests to be carried out, instead of $2^N$ as it is done in conventional spectral methods.

- Fourth, the adopted PCA formalism of the proposed method allows a set of well-established tools from the principal component analysis field, such as Bartlett's test for dimensionality[46], to be used for determining the best possible number of projection axes necessary for matching graphs that are dramatically different in size and/or structure.

In addition to these, the proposed method deploys a new correspondence clustering procedure for recovering vertex and subgraph associations, which is described in detail in the following section.

## 2.3 Clustering of projections

The proposed eigenvector subspace method allows us to determine the overall similarity of a pair of graphs by the positioning of the vertex projections of both graphs relative to each

[3]That is, although the individual eigenvectors are scaled, the projection vectors of these eigenvectors are normalized.

other. The only remaining step for solving the graph matching problem is to find the correspondence among the vertices that have similar relational properties. The main advantage of using clustering to solve this problem is, as we have seen from an earlier discussion of the clustering-based correspondence recovery methods in the introductory section, that it can equally well discover correspondence relationships of various types. That is, a clustering-based technique is no longer limited to finding the best one-to-one matches of vertices from one graph to the other, but can also identify the whole sub-graphs and vertex groups that possess similar structural properties. As such, this quality can be very important when the two graphs being matched have substantially different number of vertices, in which case a reasonable one-to-one vertex mapping may not exist.

In order to realize this, an appropriate clustering procedure has to be chosen considering the following general adequacy requirements. As stated in [41], the clustering method must be stable under growth (i.e., a resulting cluster structure is unlikely to change drastically when more objects are added), robust (i.e., small error in the description of objects lead to small changes in the clustering decomposition) and independent of the initial ordering of the objects to be clustered. According to [41], among the most popular clustering algorithms that satisfy the above criteria, the agglomerative hierarchical clustering (AHC) methods are suitable. On top of that, as we will demonstrate later in this section, a standard implementation of an AHC algorithm is the one easily adapted for the task of correspondence clustering that requires the procedure to be aware of two classes of objects being clustered.

Yet, before providing a detailed description of the clustering algorithm for graph matching capable of correspondence recovery, let us briefly consider a typical implementation of an AHC method in order to highlight several important stages of operation of the entire family of the AHC techniques. A pseudocode sequence of one such algorithm is given in Algorithm script 1.

As shown in the script, a standard AHC algorithm begins by placing each of the data objects into a separate cluster. Then, the procedure searches for a pair of clusters that are most similar to each other, i.e. the distance between such objects is smallest, and groups them into one cluster. At the same time, the distances between the remaining clusters and the newly created one are updates according to the largest (for a complete linkage method), smallest (for a single linkage method), or average (for average linkage method) distance to

**Algorithm 1** A standard agglomerative hierarchical clustering (AHC) algorithm

$N \Leftarrow$ number of objects to be clustered
**for** $i = 1$ to $N$ **do**
  assign each object $V_i$ to a separate cluster $C_i$
**end for**
**repeat**
  $[x, y] \Leftarrow \arg\min_{\forall i,j; i \neq j} distance(C_i, C_j)$
  group clusters $C_x$ and $C_y$ together
  update distances
**until** stopping criteria are met

the objects within the newly created cluster. The routine repeats until certain stopping criteria are met, such as, for instance, the desired number of clusters have been created, a given intra-cluster distance threshold is reached, etc.

On one hand, the standard AHC algorithm seems quite appropriate for correspondence clustering since it can operate directly on the vertex projections and distances between them. On the other hand the conventional AHC procedure implementation is unable to distinguish among projections of vertices that belong to different graphs, and thus cannot be geared more towards forming clusters that group together vertex projections from both graphs as opposed to clustering vertex projections belonging to the same graph. In order to make the algorithm aware of the two classes of objects involved in the clustering we introduce an extra dimension into the projection data. Depending on which graph the vertex projection belongs to, this extra dimension is assigned either 0 or a certain constant value $H$, which in effect positions vertex projections into two separate hyperplanes (see the illustration provided in Figure 2.3). With this arrangement in place, the choice of a

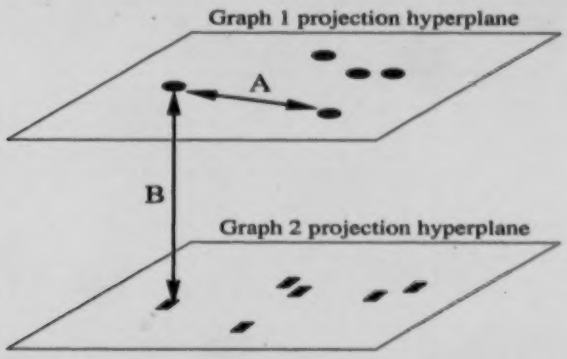



Figure 2.3: Clustering of graph eigenspace projections located in separate hyperplanes. The shown distances A and B provide an example of intra-graph and inter-graph distances, respectively.

sufficiently large value of $H$ ensures that all of the inter-graph vertex projection distances (i.e., the distances between vertex projections from different graphs) are much closer to $H$ than any of the intra-graph vertex projection distances (i.e., the distances between vertex projections from the same graph). Numerically, when all of the vertex projections are placed in their appropriate hyperplanes, a small distance (close to zero) between two projections will indicate structural similarity of a pair of vertices that come from the same graph, while a distance that is only slightly different from $H$ will signal structural similarity of a pair of vertices that belong to different graphs. Of course, one should be extremely careful when choosing the magnitude of the hyperplane separation constant, $H$. This value should always be larger than the maximum vertex projection distance for both graphs. This requirement, however, is effortlessly met in the proposed method, since the use of projection normalization procedures ensures that all of the projection coordinates lie within $[-1, 1]$ range, and, therefore, the maximum intra-graph vertex projection distance cannot be larger than 2. This leads to a conjecture that any value of $H >> 2$ can be considered valid[4].

Taking into account the above discussion, we may modify the standard AHC algorithm so as to make it suitable for graph matching as shown in Algorithm script 2. In this case, the

**Algorithm 2** A modified AHC algorithm for correspondence recovery in graph matching

$N \Leftarrow$ total number of vertices to be clustered
**for** $i = 1$ to $N$ **do**
  place vertex $V_i$ into an appropriate hyperplane
  assign vertex $V_i$ to a separate cluster $C_i$
**end for**
**repeat**
  $D_{inter} \Leftarrow \min_{\forall i,j;i \neq j} |distance(C_i, C_j) - H|$
  $D_{intra} \Leftarrow \beta \cdot \min_{\forall i,j;i \neq j} distance(C_i, C_j)$
  **if** $D_{inter} < D_{intra}$ **then**
    $[x, y] \Leftarrow \arg\min_{\forall i,j;i \neq j} |distance(C_i, C_j) - H|$
  **else**
    $[x, y] \Leftarrow \arg\min_{\forall i,j;i \neq j} distance(C_i, C_j)$
  **end if**
  group clusters $C_x$ and $C_y$ together
  update distances
**until** stopping criteria are met

clustering procedure begins by assigning each vertex to a separate cluster, much in the same way is it was done in the algorithm's original version. In addition to that, however, now

[4]The practical implementation of the algorithm, however, relies on squares of distances, rather than the distances themselves, in order to avoid the "round-off"-like effect of the square root operation, which in essence is just a numerical caveat that conceptually makes no difference with respect to the decribed above hyperplane separation approach.

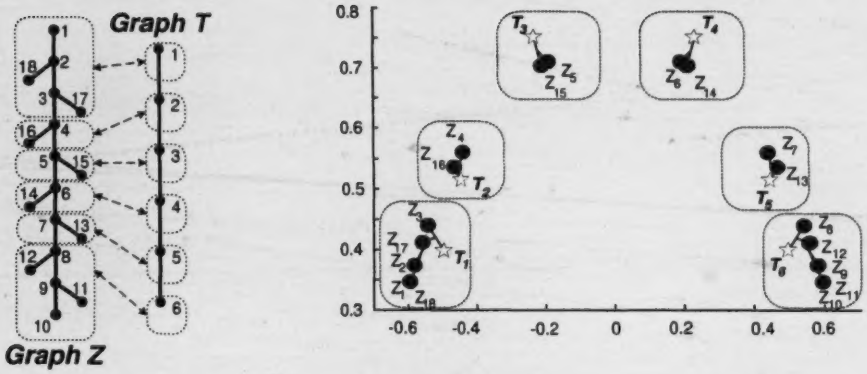
the routine adds an extra dimension to the graph vertex projection data, so that the two sets of vertices are placed into two separate hyperplanes. Then, instead of searching only for two objects, or vertices, that are most similar to each other, the procedure now looks for two pairs of objects that represent the best match among the vertices belonging to the same graph (the first pair) and among the vertices from different graphs (the second pair). It should be noted that now the clustering procedure is able to differentiate between inter-graph and intra-graph vertex projection distances judging by their magnitudes: the smallest in the absolute value distance would correspond to the best matching between vertices from the same graph, while the distance that differs least from the value of hyperplane separation constant $H$ would be the best match between vertices that come from different graphs (see the symbolic calculations of $D_{inter}$ and $D_{intra}$ in Algorithm script 2).

At this point, any appropriate logic can be applied to make the clustering algorithm treat selectively best inter-graph and intra-graph candidates for clustering. Currently, the developed algorithm dynamically switches clustering regimes (i.e., intra-graph vs. inter-graph vertex clustering) according to the type of the best match at hand, and proceeds as specified in its standard version. One can also notice from the listing provided in Algorithm script 2 that in the present implementation the intra-graph distance values are conditioned by a certain preference factor $\beta$. This parameter $\beta$, if set to a value greater than 1, makes the algorithm more inclined, or biased, towards clustering the vertices belonging to different graphs, and thus, induces the procedure to grow clusters around the best inter-graph vertex correspondences. For any particular application, an appropriate value $\beta$ may be estimated by training.

Finally, there is a necessary modification in the stopping criteria for the presented correspondence clustering algorithm, since neither the desired number of clusters nor any reasonable distance threshold are known for certain in advance. According to its present implementation, the algorithm stops as soon as all of the vertex projections have been associated with a certain cluster and there are no more (if at all) "orphan clusters"[5] left.

In order to illustrate the above described correspondence clustering procedure, let us consider the example depicted in Figure 2.4. As shown, Figure 2.4 demonstrates the result

[5]There are situations when several distant clusters are found to contain vertices that belong to one graph only. Thus, for vertices in such clusters, there are no corresponding vertices from the other graph, which is why the clustering procedure needs to keep running in order to find other suitable clusters to attach the "orphans" to.




(a) Graphs Z, T.

(b) Clustered projections of graphs Z and T (the formed clusters of vertex correspondence are circled).

Figure 2.4: Example 2: Clustering of vertex projections of sample graphs Z and T.

of vertex projection clustering (Figure 2.4(b)) of two sample graphs $Z$ and $T$ with 18 and 6 vertices respectively, that recovers a natural correspondence among the groups of vertices in these two graphs (Figure 2.4(a)). The linkages that were established between vertices during the clustering phase are shown as straight lines connecting the projected graph vertices and/or midpoints of other lines representing correspondence clusters (see Figure 2.4). The resulting cluster decomposition obtained from correspondence clustering of the two graphs $Z$ and $T$ is summarized in Table 2.1.

| Cluster number | Clustered vertices from graph T | Clustered vertices from graph Z |
|---|---|---|
| 1 | 1 | 1, 2, 3, 17, 18 |
| 2 | 2 | 4, 16 |
| 3 | 3 | 5, 15 |
| 4 | 4 | 6, 14 |
| 5 | 5 | 7, 13 |
| 6 | 6 | 8, 9, 10, 11, 12 |

Table 2.1: Cluster decomposition obtained from correspondence clustering of the two graphs $Z$ and $T$

Once the correspondence clustering decomposition has been obtained, the only remaining task is to provide a numeric estimate, or a measure, of how similar the two graphs being compared are to each other. For this pupose, a similarity metric based on the quality of the performed correspondence clustering should be a well-grounded choice, since one may

generally judge how similar one graph is to another by looking at how well their projected vertices cluster together. In addition to that, the sought similarity estimate should favour greater number of balanced correspondence clusters, in order to have the property of reaching a maximum when matching the same graphs. Taking these issues into consideration, the proposed method uses a graph similarity measure defined as follows:

$$Sim_{Graph} = \alpha_1 \cdot Sim_{DB} + \alpha_2 \cdot Sim_{set} \tag{2.6a}$$

$$Sim_{DB} = \frac{1}{M} \cdot \sum_{i=1}^{M} \max_{j=1..M; j \neq i} (d_{ij}), \quad where \quad d_{ij} = \frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \tag{2.6b}$$

$$Sim_{SET} = \frac{\sum_{i=1}^{M} (1 - p_i)}{rank(A_{G_1}) + rank(A_{G_2})}, \quad where \quad p_i = \frac{|V_i^{G_1} - V_i^{G_2}|}{V_i^{G_1} + V_i^{G_2}} \tag{2.6c}$$

As Equation 2.6a shows, the proposed similarity metric attempts to strike an appropriate balance by being essentially a combination of qualitative ($Sim_{DB}$) and quantitative ($Sim_{SET}$) correspondence clustering assessment measurements. The weighted sum format of the formula allows for a larger degree of flexibility, since its actual parameters ($\alpha_1$ and $\alpha_2$), may be adjusted in accordance with the requirements of a particular application using training techniques.

The first component in the proposed graph similarity formula is the well-known Davies-Bouldin (DB) cluster validity index, calculated as specified in Equation 2.6b. Here, $M$ is the number of vertex correspondence clusters, $\sigma_i$ is the average distance of all vertex projections in cluster $i$ to their cluster center $c_i$ and $d(c_i, c_j)$ is the distance between cluster centers $c_i$ and $c_j$. The index values range within the interval of $[0, \infty]$ with smaller values corresponding to a good clustering, which necessitates the value of $\alpha_1$ to be negative.

The second component is an estimate (Equation 2.6c) derived from the idea of the set similarity formula (the "intersection-over-union" metric), that encourages greater number of the discovered correspondence clusters, while at the same time introduces a penalty factor $p_i$ dependent on the degree of how unbalanced cluster $i$ is. In the presented formula, $M$ is the number of vertex correspondence clusters, $V_i^{G_1}$ and $V_i^{G_2}$ are the number of vertices from graph one and graph two, respectively, found in $i$-th correspondence cluster, and $A_{G_1}$, $A_{G_2}$ - the adjacency matrix representations of the two graphs to be matched.

# Chapter 3

# Application one: matching of 2D shapes represented by shock graphs

## 3.1 Shock graph representation

The first application to be considered is the task of matching two-dimensional shapes represented by shock graphs. The technique used for deriving shock graph representations originates from the singularity theory and is based on the idea of applying a curve evolution process to a given shape which leads to formation of a number of entropy-satisfying singularities, or shocks (for more details, please see [34, 8, 33]). The locus of the obtained shock positions gives the so called Blum's medial axis that represents a rough skeletal structure of the shape in question and has the property of preserving the extent and connectivity of its various regions. Then, the behavior of a radius function alongside the extracted medial axis is examined in order to classify shocks into four different categories. A shock is said to be of type 1 whenever the radius function varies monotonically (this corresponds to protruded regions of the original shape), type 2 if the radius function achieves a strict local minimum (shocks of this type occur in neck-shaped regions), type 3 when the radius function is constant along an interval (for bend-like regions with parallel sides), and type 4 if the radius function achieves a strict local maximum (for regions in the original shape that evolve into isolated spots). Figure 3.1 gives an illustration of shocks of these four distinct categories.

Finally, the above 4-type classification combined with the temporal relational information derived from shock formation time data provides a complete framework for representing shapes as attributed DAG's. It's important to note that, as it has been shown in [34], any

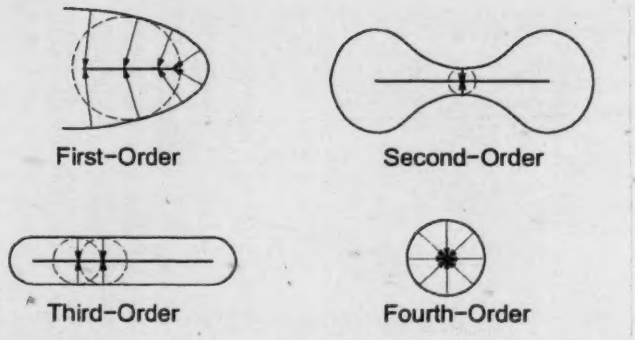


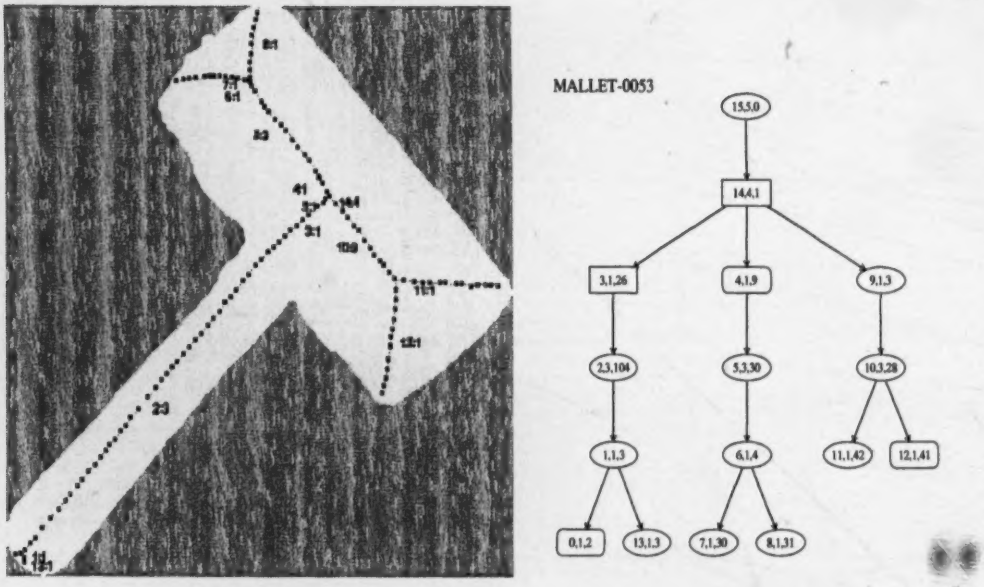

Figure 3.1: Four schematic shape regions and their corresponding shock types (adapted from [34])

given two-dimensional shape has a unique shock graph corresponding to it, which is a valuable property of the representation for solving shape matching problem via graph matching. An example of a two-dimensional shape and its shock tree representation is depicted in Figure 3.2.

## 3.2 Experimental results

In order to evaluate the performance of the proposed method for the task of shape matching, a medium-sized database of shock graphs derived with the above described technique from a set of images was used. Every shape in the database served as a query and was compared against all of the other images. For each such comparison, a similarity value was calculated as specified in section 2.3. The results of these experiments have shown that for the majority of shape prototypes the appropriate instances are retrieved within 5 top matches or better. The summary of the similarity calculations among the first 23 two-dimensional shapes belonging to various classes is provided in Figure 3.3[1]. The table that the figure shows has its topmost row representing prototype images of various shape categories, and the leftmost column depicting a number of shape instances belonging to some of the shown categories from the topmost row. A number found at the intersection of a given category column and a shape instance row is the measure of similarity between the corresponding shape instance and category prototype. As it can be seen from the data provided in Figure 3.3, the proposed method performs accurately by recovering the correct correspondence between

[1]The similarity values in this table are not normalized to a [0,1] interval. Also, most of the values marked as 0.00 are different from zero, but not large enough to be represented within the first two decimal digits of precision.




(a) A sample shape

(b) Shock graph representation of a shape (first two vertex numbers correspond to labels in Figure 3.2(a))

Figure 3.2: A sample shape of a mallet and its shock graph representation

shape instances and prototypes of different categories.

In other experiments, in which the shapes belonging to the same category were compared against a category prototype image, the method has also shown reasonable results. An example of one such experiment is given in Figure 3.4, which depicts a prototype image of a boxer dog category on the left and lists the shape instances of this category in the order of the calculated similarity on the right.

Among the important features of the proposed shape matching method are its abilities to discriminate well among objects from different categories, provide a partial ordering of similar shapes, and deal with a certain degree of occlusion, distortion and noise. These properties, however, have already been attained in some of the previously developed solutions for the shape matching problem. For instance, the procedure documented by Dickinson et al. [32] uses topological signature vectors, derived from the eigendecompositions of subtrees underlying any given vertex in a shock graph, together with an enhanced version of depth-first search achieving comparable results. Also, Pelillo et al. [26] have developed a technique for shape matching via a method based on finding a maximum clique in an association graph

structure. The authors' method takes into account specific properties of the shock graphs, and thus is especially suitable for the application of shape matching. With respect to performance, both of the above methods and the one proposed in this paper are similar in the sense that they all demostrate encouraging results[2] of high accuracy when applied to the task of matching shapes represented by shock graphs.

There is, however, an important distinction between the proposed method and what has been done before. Namely, it is the valuable property of the proposed method of being able to establish structural correspondence among groups of vertices and subgraphs, in addition to the one-to-one vertex correspondence, with the aid of clustering. That is, not only does the proposed method provide a numeric estimate of how similar two graphs are to each other, but it also allows one to determine which vertices, subgraphs, and vertex groups of one graph correspond to those of the other graph via the obtained cluster decomposition. It is important to note that deriving this structural correspondence information does not involve any additional computations. Indeed, as it was shown in section 2.3, clustering is performed as a required step of graph similarity estimation, and therefore all of the structural correspondence data obtained from the derived cluster configuration may be considered a useful by-product of the procedure.

In order to illustrate this property of recovering correspondence through clustering, let us consider two sample shapes together with their respective shock graph representations depicted in Figure 3.5. The two shapes shown in Figures 3.5(a) and 3.5(b) belong to a common prototype of a boxer breed dog, and visually are quite similar to each other. Moreover, one may conjecture that the first image is a copy of the second one which has undergone a number of transformations, such as rotation, change in scale, occlusion (e.g., front leg, tail). These transformations, in turn, are reflected in the shock graph representations of both shapes (e.g., a more stretched "torso" subgraph in boxer-10 graph, a different attachment positions of "back" and "hind legs" subgraphs in boxer-24 shock graph, distortions in "head" subgraphs, etc.), which makes the two candidate shock graphs non-isomorphic and only approximately similar to each other. Nevertheless, the proposed method is able to identify and group into clusters vertices and subgraphs according to their structural similarity, while the obtained cluster decomposition, for the chosen example, is both intuitively understood and

[2]even though these results may not be strictly comparable, since some of the methods use vertex type information in addition to the relational data provided by a graph's structure.

easy to judge, since most of the derived correspondence clusters make sense from the point of view of dog anatomy (i.e., the vertices that represent the head of one dog are clustered together with those that correspond to the head of the other dog, etc.) The resulting cluster decomposition obtained from correspondence clustering of the two shock graphs for shapes boxer-10 and boxer-24 is summarized in Table 3.1. Given the data provided in table 3.1, one

| Cluster number | Clustered vertices from shape BOXER-24 | Clustered vertices from shape BOXER-10 |
|---|---|---|
| 1 | 2(back) | 3(back), 11(back), 12(back) |
| 2 | 27(torso), 29(head), 30(head) | 13(torso), 34(head), 36(head), 37(head), 2(back) |
| 3 | 28(head), 32(head) | 33(head), 35(head) |
| 4 | 5(hind legs), 12(hind legs) | 7(hind legs), 8(hind legs) |
| 5 | 15(front legs), 16(front legs), 6(hind legs), 8(hind legs) | 26(front legs), 31(front legs) |
| 6 | 3(hind legs), 4(hind legs), 7(hind legs), 11(hind legs) | 4(hind legs), 5(hind legs), 6(hind legs), 9(hind legs) |
| 7 | 17(front legs), 18(front legs), 19(front legs), 9(hind legs), 10(hind legs) | 28(front legs), 29(front legs), 30(front legs |
| 8 | 1(torso), 34(back) | 1(torso), 17(back) |
| 9 | 13(torso), 14(front legs), 20(front legs), 21(front legs), 22(front legs), 23(torso), 24(torso), 25(neck), 26(neck) | 15(torso), 16(torso), 22(neck), 23(neck), 24(front legs), 25(front legs), 27(front legs), 32(front legs) |
| 10 | 31(ears), 33(ears) | 10(tail), 14(tail) |
| 11 | 35(back), 36(back) | 18(torso), 19(back), 20(back), 21(back) |

Table 3.1: Cluster decomposition obtained from correspondence clustering of the two shock graphs for shapes BOXER-10 and BOXER-24. Each cluster enumerates the vertices from the two graphs that are grouped together and lists the part of the body that the vertex approximately comes from, in brackets.

may notice that the correspondence relationships recovered from clustering decomposition are of either one-to-many or many-to-many type. These correspondence types are generally not discovered by the conventional graph matching techniques, which are mainly capable of finding best one-to-one vertex associations only. It also should be mentioned, that all of the reported above experiments were purely structural; that is, for the purpose of matching shock graphs the proposed method relied exclusively on the shock graph relational data, without taking into account any other information.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.52 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.26 | 0.10 | 0.00 | 0.00 |
| | 0.00 | 5.00 | 0.00 | 0.00 | 2.18 | 0.66 | 0.00 | 0.00 | 0.00 | 0.59 | 0.00 |
| | 0.00 | 0.00 | 3.35 | 5.00 | 0.00 | 1.40 | 0.00 | 0.00 | 0.00 | 0.30 | 0.00 |
| | 0.00 | 0.00 | 5.00 | 3.35 | 0.06 | 0.63 | 0.00 | 0.00 | 0.00 | 2.93 | 0.00 |
| | 0.00 | 1.39 | 0.00 | 0.20 | 2.34 | 0.87 | 0.00 | 0.08 | 0.00 | 0.22 | 0.00 |
| | 0.47 | 0.99 | 0.00 | 0.24 | 1.41 | 2.61 | 0.00 | 0.00 | 0.62 | 0.20 | 0.00 |
| | 0.52 | 0.67 | 0.63 | 1.40 | 0.40 | 5.00 | 0.00 | 0.00 | 0.00 | 0.54 | 0.00 |
| | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.26 | 0.20 | 0.00 | 0.00 | 0.03 |
| | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.37 | 0.00 | 0.00 | 0.00 |
| | 0.46 | 0.02 | 0.00 | 0.00 | 0.04 | 0.20 | 0.00 | 0.01 | 2.55 | 0.00 | 0.00 |
| | 0.00 | 0.59 | 2.93 | 0.30 | 0.13 | 0.54 | 0.00 | 0.00 | 0.00 | 5.00 | 0.00 |
| | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.15 | 0.00 | 0.00 | 0.00 | 0.67 |

Figure 3.3: Shape similarity summary table as computed according to the formula in Equation 2.6a (a box drawn around a similarity value corresponding to best matching pair)

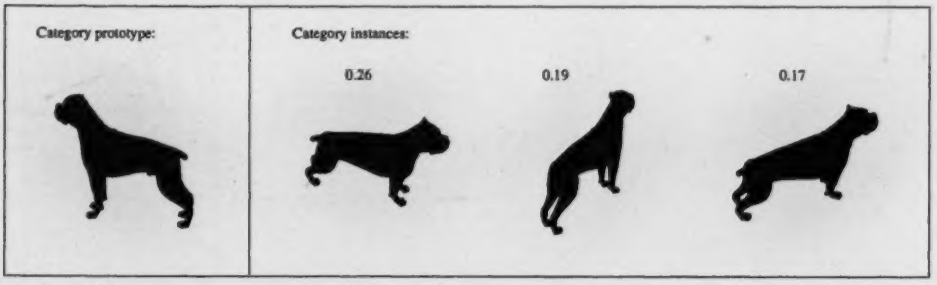
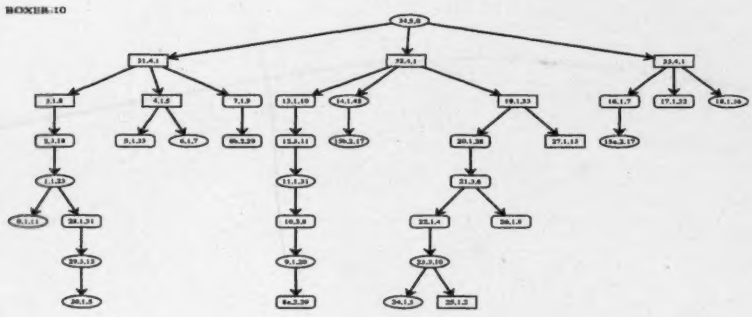


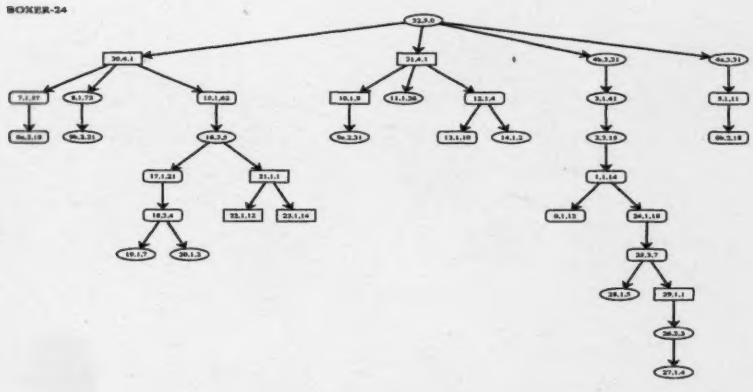

Figure 3.4: An example of comparison of shapes within the same category (the calculated similarity values are shown above each shape instance)

(a) Shape: BOXER-10

(b) Shape: BOXER-24




(c) Shock graph representation of BOXER-10




(d) Shock graph representation of BOXER-24

Figure 3.5: Two sample shapes BOXER-10 and BOXER-24, and their shock graph representations

# Chapter 4

# Application two: information retrieval with natural language processing enhancements

## 4.1 General information retrieval problem setting

For the subdiscipline of information retrieval that deals with textual data, the main concern is to be able to establish the degree of relevance of a given document to a query. When a set of documents and queries is considered, an information retrieval system is required to rank the documents in accordance with their relevance to a particular query, presenting a certain number of the best matching documents as an answer set.

Of course, in order to be able to render a numeric relevance estimate for documents and queries, the ranking procedure must rely on a specific common representational model. One such model, usually referred to as *vector model representation*, is used in the majority of state of the art textual information retrieval systems. The model represents uniformly both documents and queries as vectors of weights of a predetermined number of important words, called keywords or index terms. The actual values of the index term weights are calculated with respect to the occurence information of the corresponding keywords in a particular document or query. Once all of the weights for documents and queries are computed, a similarity measure is assigned to each document and query pair. It is the actual value of this similarity measure that is used to determine whether a document is relevant to a query and rank the entries in the answer set accordingly.

In order to illustrate the essential principles of functioning of a typical keyword-based information retrieval system that uses vector model representation, as well as to provide a

more clear explanation of how it may be improved by incorporating several techniques that the proposed graph matching method is capable of, let us consider a simple example of the retrieval process for a mini-set of documents, shown in Table 4.1.

| *Number* | *Document* |
|---|---|
| 1 | The UNIX Programming Environment |
| 2 | The C Programming Language |
| 3 | The Art of Computer Programming |
| 4 | Women in Art |
| 5 | History of Western Art |

Table 4.1: A mini-set of documents consisting of several book titles

The above set of documents represents 5 book titles, some of which are about computer programming, while others are more related to the subject of art. Suppose that the following set of words occuring in the above documents is chosen to be index terms[1] : *UNIX(1), Programming(2), Environment(3), Language(4), Art(5), Computer(6), Women(7), History(8), Western(9)* . Given these keywords, the documents in our small book title dataset can now be cast into the vector representation framework, where each keyword corresponds to a single vector component whose value is set according to a certain function of the index term occurence frequency[2]. In the simplest case, the vector weights are set to a given index term's frequency, i.e. the number of times a certain keyword occurs in a document (or query). For instance, the second document from the above mini-set contains one keyword number 2 (*"Programming"*) and one keyword number 4 (*"Language"*), hence the vector representation of this document will be $d_2 \leftarrow \{0,1,0,1,0,0,0,0,0\}$. In the same way, the rest of the documents in the dataset may be reprsentd as vectors, as summarized in Table 4.2. Similarly, a query can also be provided with a vector representation. For example, a sample query such as *"Find books about computer programming"* will be represented as vector $q \leftarrow \{0,1,0,0,0,1,0,0,0\}$.

Once the document and query vectors have been created based on whatever weighting scheme is selected, they can be used to compute the degree of similarity between the cor-

[1]In general, the process of determining which words should be selected as index terms is quite complex, since it usually needs to take into account the frequency of occurence of a certain word, eliminate a number of specific low content words, known as stop words, etc. For the sake of the prsented example, this, however is not an essential issue.

[2]The actual process of calculating term weights usually involves such operations as normalization over the total number of index term occurences, smoothing, and weight adjustment according to a keyword's document discriminatory power introduced with the inverse document frequency (IDF) factor, but, again, for the sake of the present simplified example, these details are not important.

| *Index term* | *Documents* | | | | |
|---|---|---|---|---|---|
| | $doc_1$ | $doc_2$ | $doc_3$ | $doc_4$ | $doc_5$ |
| UNIX | 1 | 0 | 0 | 0 | 0 |
| Programming | 1 | 1 | 1 | 0 | 0 |
| Environment | 1 | 0 | 0 | 0 | 0 |
| Language | 0 | 1 | 0 | 0 | 0 |
| Art | 0 | 0 | 1 | 1 | 1 |
| Computer | 0 | 0 | 1 | 0 | 0 |
| Women | 0 | 0 | 0 | 1 | 0 |
| History | 0 | 0 | 0 | 0 | 1 |
| Western | 0 | 0 | 0 | 0 | 1 |

Table 4.2: Vector representation of documents from the book title dataset (each document is a column vector)

responding documents and queries. This correlation is typically obtained by computing the cosine of the angle between a document and a query vector, as expressed by Equations 4.1a (in vector form) and 4.1b (in weight coefficient form):

$$Vectors: \quad Sim_{COS}(doc_i, q) = \frac{\overrightarrow{doc_i} \times \overrightarrow{q}}{|\overrightarrow{doc_i}| \cdot |\overrightarrow{q}|} \tag{4.1a}$$

$$Weights: \quad Sim_{COS}(doc_i, q) = \frac{\sum_{k=1}^{N} w_k^{doc_i} \cdot w_k^q}{\sqrt{\sum_{k=1}^{N} \left(w_k^{doc_i}\right)^2} \cdot \sqrt{\sum_{k=1}^{N} \left(w_k^q\right)^2}} \tag{4.1b}$$

where $N$ is the number of index terms used, $w_k^{doc_i}$ and $w_k^q$ are the weights of $k$-th index term in an $i$-th document and a query, respectively. The possible values of the Cosine similarity measure are confined to the $[0, 1]$ range, which is a convenient property of the metric that allows for a uniform document relevance ranking. For the above example of a sample query (*"Find books about computer programming"*) and the set of five book title documents, applying the Cosine similarity measure produces the following results (see Table 4.3). As expected, the resulting ranking favours the first three book titles from the document

| *Number* | *Document* | $Sim_{COS}(doc_i, q)$ | *Rank* |
|---|---|---|---|
| 1 | The UNIX Programming Environment | 0.408 | 3 |
| 2 | The C Programming Language | 0.500 | 2 |
| 3 | The Art of Computer Programming | 0.817 | 1 |
| 4 | Women in Art | 0.000 | 4 |
| 5 | History of Western Art | 0.000 | 5 |

Table 4.3: Similarity measures and the resulting ranking for a sample query and documents from the book title dataset

set that are indeed related to computer programming, hence the returned answer set can be considered correct.

Although generally regarded as analytically elegant and practically useful, the keyword vector representation model has an important disadvantage: once words are converted into the index terms, all of the information about their context and syntactic relations to other words in a document or query is completely lost[3]. In order to deal with this problem, various methods have been developed, most of which focused on improving the retrieval performance by taking into account an index term's context information derived from various word co-occurence statistics [1, 44]. There is, however, another alternative way to attempt to make an information retrieval system more "intelligent" and boost its performance, as suggested in [35, 42]. Namely, one may incorporate the information about syntactic relations among keywords in the process of matching documents and queries in order to improve the retrieval results. And this is exactly where the proposed method for matching graphs and generic relational structures may be applied - as described in the following section.

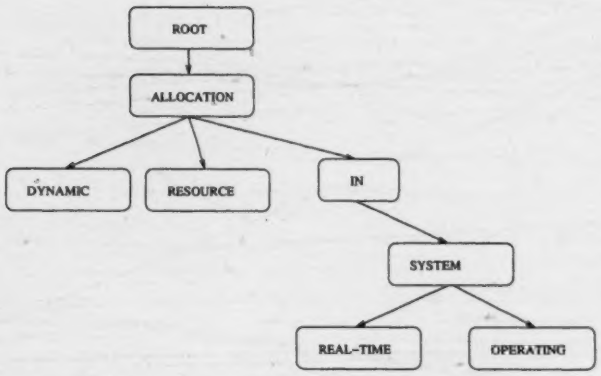
## 4.2 Enhancing information retrieval with syntactic relevance data

As it has been mentioned before, the standard vector model for information retrieval discards all of the relational information imposed by the syntactic structure of sentences that constitute documents and queries. In order to illustrate this drawback, let us consider a representative example as formulated in [24]. Suppose that a certain information retrieval system operates only on three index terms $t_1$, $t_2$, $t_3$ that correspond to the words *operating*, *system*, *dynamic*, respectively. Assume, also, that two entirely different queries are submitted as input for such a system: the first, $q_1$, inquiring about *dynamic resource allocation in a real-time operating system* and the second, $q_2$, that is concerned with *dynamic system models of information agents operating on the internet.* One can clearly see that all of the index terms $t_1$, $t_2$, $t_3$ are present in $q_1$ and $q_2$, which makes the two queries indistinguishable for the information retrieval system in question. Intuitively, it appears reasonable to try to improve the document-query matching process by adjusting a weight of a given index term match with respect to its syntactic relevance, which requires that a certain procedure capable of comparing the degree of syntactic similarity of a pair of matching keywords be

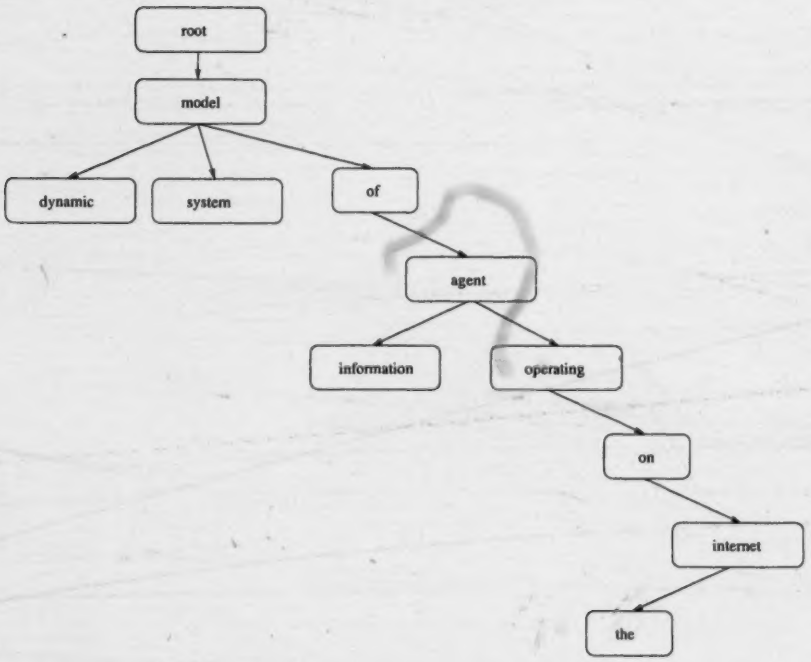
[3]In the above example, for instance, a query on the topic of art (single keyword) will produce the result that implies that both book 3 ("The Art of Computer Programming") and book 5 ("History of Westrn Art") are equally related to the subject of art, which is hardly true

designed.

For this pupose, the first task to be addressed is deriving a graph representation of queries and documents in the form of term dependency trees. In the current implementation, the *Minipar* syntactic parser developed by Lin[16] is used. Applying the parser to queries $q_1$ and $q_2$ from the above example, the following graph representations of syntactic relations can be derived (see Figure 4.1).




(a) Parse tree of query $q_1$




(b) Parse tree of query $q_2$

Figure 4.1: Graph representation of syntactic relations among the words in sample queries $q_1$ and $q_2$

This illustration of query parse trees provides some important information. First, it confirms the fact that the syntactic roles that index term $t_3$ (*dynamic*) plays in both queries is most similar. Indeed, one may see that in queries $q_1$ and $q_2$ index term *dynamic* is used to modify the head noun of each query. Second, the syntactic roles of index term $t_1$ (*operating*) in the two queries are also quite similar, since in both cases this keyword modifies a noun attached to the query head noun via a preposition, even though there is a slight difference in syntactic context[4]. Third, it is apparent that the syntactic function of index term $t_2$ (*system*) in query $q_1$ differs most from its functional purpose in query $q_2$: in the former query the keyword is a content noun related to the head noun via a preposition, while in the latter it is merely a head noun modifier. Summing up these findings, it can be stated that the comparison of the roles of the selected index terms with respect to the syntactic relations among the words in the two queries leads to a conclusion that the keyword match on index term $t_2$ (*system*) in queries $q_1$ and $q_2$ is far less syntactically relevant than those on index terms $t_1$ (*operating*) and $t_3$ (*dynamic*). This is about as much additional information as one can get from the syntax alone, without considering the semantics, co-occurence statistics or any other parameters of the words used. As little as it may seem to be, this additional piece of knowledge may signal an irrelevant match for the vector model, which would otherwise be unable to distiguish the two queries at all.

In order to be able to draw such conclusions automatically and, subsequently, incorporate the feature in the operation of a typical keyword-based information retrieval system, we may use the proposed graph matching method. Recall that an important property of the method is such that structurally similar vertices are projected into the regions that are close to each other. Therefore, by performing a matching between document and query term dependency graphs, one can judge the degree of structural similarity and, hence, syntactic relevance of a given pair of words by looking at how far or how close the corresponding vertices are in the projected space. To provide an illustration to this perspective, let us again use queries $q_1$ and $q_2$ from an earlier example. Figure 4.2 shows the result of projecting parse trees of these two queries as performed by the proposed graph matching method. In this diagram, the projections of the words from query $q_1$ are shown as filled blue circles, while the projected terms of query $q_2$ are depicted as red diamonds. Also, the projection space distances between

[4]Namely, in the parse tree of query $q_1$ keyword *operating* is a terminal node, while in $q_2$ it is linked to a number of other modifying terms via preposition *on*.

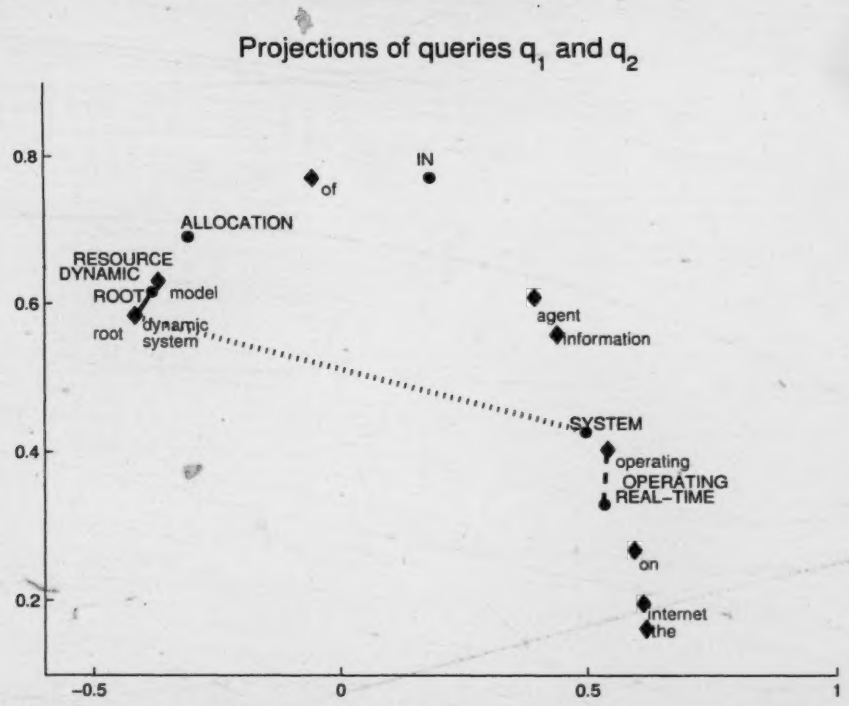



Figure 4.2: Eigenspace projections of the parse trees of queries $q_1$ and $q_2$ (the following lines are drawn in order to visualize the distances between projections of the matching index terms: a solid line for keyword *dynamic*, a dashed line for keyword *operating*, and a dotted line for keyword *system*)

the corresponding pairs of keywords $t_1$, $t_2$ and $t_3$ in the two queries are drawn as dashed, dotted and solid lines respectively.

Let us consider the distances between the projections of matching keywords from the two sample queries, which are graphed as a bar chart in Figure 4.3. Obviously, the distance between the projections of index term $t_2$ (*system*) is much larger than those between the remaining pairs of matching keywords, which, given the eigenspace projection distance interpretation imposed by the graph matching method, confirms our earlier findings: this result shows that the keyword match on word *system* has a far lower degree of syntactic relevance than the other two. And this exactly reflects the logic used in the present implementation of the modified information retrieval systems that attempts to take into account the syntactic relevance of every keyword match between a document and a query. It adheres to the following simple principle: whenever a pair of matching index terms is found to be syntactically relevant (i.e., it is projected into the regions that are close to each other), the similarity estimate assigned to the corresponding document and query pair is awarded a

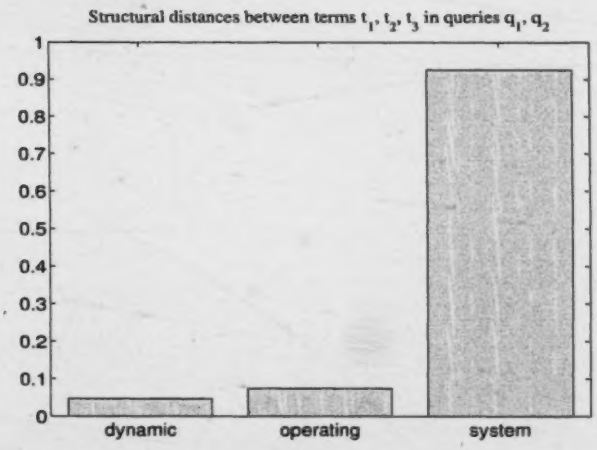



Figure 4.3: Distances between the eigenspace projections of matching keywords from the two sample queries $q_1$ and $q_2$

bonus, otherwise, it is penalized for a syntactically irrelevant keyword match.

In order to implement this feature within the framework of the vector representaton model, the similarity calculation formula (Equation 4.1b) has to be changed to accommodate the additional knowledge about syntactic relevance of each keyword match, as shown in Equation 4.2.

$$Sim_{COS+SYNTAX}(doc_i, q) = \frac{\sum_{k=1}^{N} w_k^{doc_i} \cdot w_k^q \cdot \Phi_k(doc_i, q)}{\sqrt{\sum_{k=1}^{N} \left(w_k^{doc_i}\right)^2} \cdot \sqrt{\sum_{k=1}^{N} \left(w_k^q\right)^2}} \tag{4.2}$$

This formula of similarity measure between a document and a query (expressed using the same notation as in Equation 4.1b) introduces an adjustment factor $\Phi$ that follows the logic of the mentioned above principle: reward a syntactically relevant keyword match, penalize an irrelevant one. The mathematical representation of this adjustment factor $\Phi$ is chosen to be a simple thresholding function, as expressed in Equation 4.3:

$$\Phi_k(doc_i, q) = \begin{cases} 1 + B_{SYNTAX} & \text{if } \frac{D_k(doc_i,q)}{D_{max}(doc_i,q)} < T_{SYNTAX} \\ 1 - P_{SYNTAX} & \text{if } \frac{D_k(doc_i,q)}{D_{max}(doc_i,q)} \geq T_{SYNTAX} \end{cases} \tag{4.3}$$

where $D_k(doc_i, q)$ and $D_{max}(doc_i, q)$ are the distance between projections of $k$-th keyword and the maximum distance among the projections of all of the words from document $doc_i$ and query $q$ respectively; $B_{SYNTAX}$, $P_{SYNTAX}$ are percentage reward and penalty factors, and $T_{SYNTAX}$ is the syntactic relevance distance percentage threshold, which can be properly estimated experimentally with training data.

## 4.3 Experimental results

In our experiments, we used two standard text collection ADI[39] and CISI[40]. The two collections have the following parameters (see Table 4.4). For each text collection, we com-

| | *ADI collection* | *CISI collection* |
|---|---|---|
| Documents | 82 | 1460 |
| Number of terms | 5491 | 189969 |
| Number of unique terms | 1693 | 19497 |
| Queries | 35 | 112 |
| Number of terms | 499 | 9679 |
| Number of unique terms | 42 | 2649 |

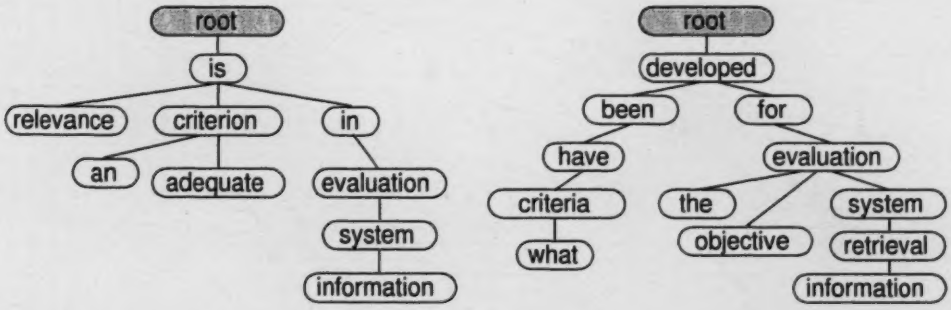
Table 4.4: Parameters of the two text collections used in experiments

pare the performance of the baseline information retrieval system with that of the modified one. The baseline system operates according to the standard vector model using $TF \cdot IDF$ index term weighting with low content stop word elimination[41], and a Porter stemming technique[18]. The modified system works with the same data as does the baseline one, but adds the syntactic relevance adjustment (factor $\Phi_k$, see Equation 4.2) of a keyword match to the document/query similarity calculation. Additionally, since the majority of documents and queries consist of several sentences, an aggregation technique is used to combine the syntactic information of each sentence derived with *Minipar* parser into a common lattice of keyword relations. This aggregate relational lattice is constructed as an adjacency matrix of a weighted graph, in which more prominent relations among keywords are assigned larger weights, and thus play a more important role during syntactic graph matching (for a detailed example of aggregate relational lattice construction, please see the Appendix).

The overall performance of the modified system does not significatly differ from that of the baseline system (for instance, modified average precision for ADI 49.6%, baseline 49.0% and similar for CISI collection), which confirms the earlier documented results[24, 35, 42]. On the other hand, individual query analysis shows that the performance of the modified information retrieval system is not uniform and is consistently better for a subclass of queries with some specific properties. Namely, the retrieval results are improved in the majority of cases when a query is short (8-16 words) and is presented as one or two sentences.

In order to illustrate a typical situation in which syntactic relevance relevance of a keyword match constributes to the retrieval performance of short queries, let us consider an

example of matching query 13 and document 27 from ADI text collection (see Figures 4.4[5], 4.5, 4.6). Although both the query and the document have a substantial keyword overlap,




(a) from Document 27: *"Is relevance an adequate criterion in retrieval system evaluation"*

(b) Query 13: *"What criteria have been developed for the objective evaluation of information retrieval and dissemination system?"*

Figure 4.4: Parse trees of sample sentences from document 27 and query 13

the baseline retrieval system does not recognize this pair as the best match. The modified information retrieval system, however, produces an improved ranking of document 27 with respect to query 13. As one can see from Figures 4.5 and 4.6, the common keywords (*criteria*, *evaluation*, *retrieval*) that the document and the query share are projected into regions that are quite close to each other. Consequently, the system rewards the syntactic relevance of the keyword match and promotes document 27 to a better rank, which results in an improvement in retrieval performance. This is the case for both of the illustrated matching technique variations: when syntatic relational information of the query is compared **(a)** with the relational structure of the most contributing sentence (Figures 4.5), and **(b)** with the whole aggregated relational lattice of document 27 that summarizes the syntactic relations among all of the terms from all of the sentences in the document (Figures 4.6), from which it may be inferred that for matching short, one sentence queries both aggregate lattice and sentence-by-sentence approaches are nearly equivalent[6].

The summary of the performance of the baseline and modified information retrieval systems for the subset of short queries is given in Figure 4.7 and Table 4.5.

Taking into account the above discussion, it can be said the obtained results are quite

[5]Figure 4.4(b) shows the simplified parse tree after conjunction expansion and prepositional post-modifier normalization.

[6]while the former, of course, would be much more computationally efficient than the latter.

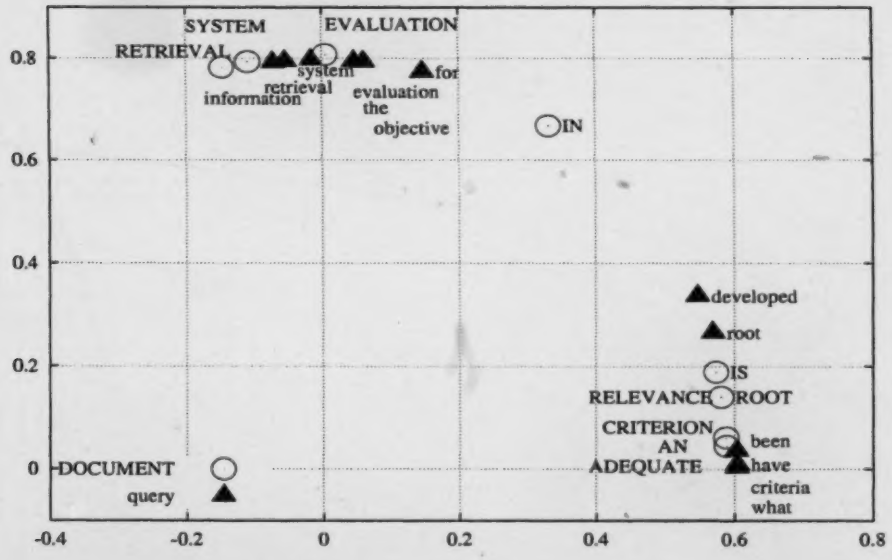



Figure 4.5: Eigenspace projection of syntactic structure of query 13 and a sample sentence from document 27 (ADI collection)

| *Text collection* | *11-point Precision Average* | | *Relative* |
|---|---|---|---|
| | *Baseline IR system* | *Modified IR system* | *% change* |
| ADI | 54.32 | 61.38 | +12.98% |
| CISI | 35.70 | 37.72 | +5.64% |

Table 4.5: Performance of baseline (B.) and modified (M.) retrieval systems for the subset of short queries from ADI and CISI text collections

encouraging and generally confirm earlier findings[24]. While the overall improvement in retrieval performance is not large, the application of the proposed graph matching method was shown to be beneficial to a certain subclass of queries, for which the syntactic relational data was deemed as important as the semantics of the matching keywords themselevs. When judging the quantitative effect achieved by augmenting a standard IR system with syntactic information, one needs to keep in mind that the discipline of general IR is mostly concerned with the retrieval of relevant content that is in most cases unequivocally conveyed by the appropriately selected index terms, hence the marginal magnitude of the obtained improvement. On the other hand, one may expect that for the problem settings where relational information is an essential and indispensable issue (such as question-answering systems, natural language interfaces to relational databases, etc.), the potential for performance improvement is more than adequate.

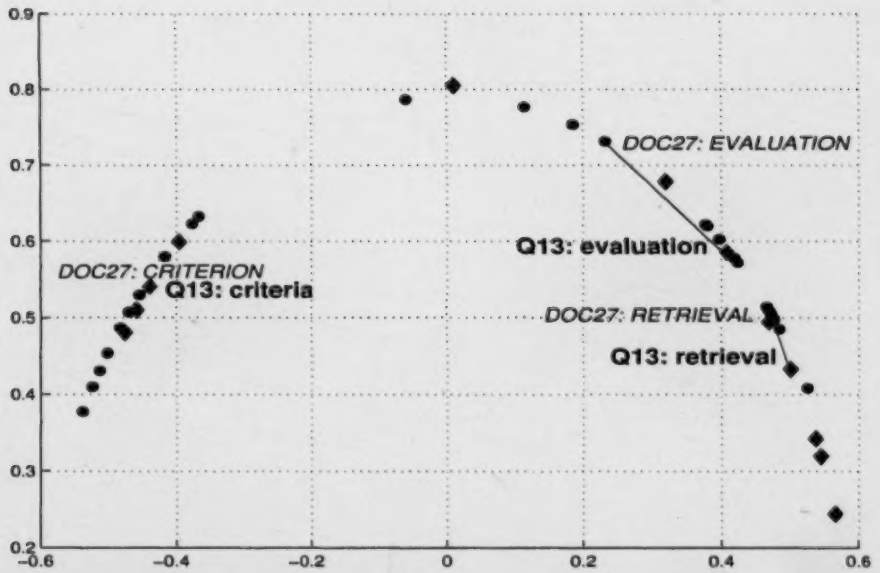



Figure 4.6: Eigenspace projection of syntactic structure of query 13 and aggregated syntactic relational lattice of document 27 from ADI collection (query keywords are depicted as red diamonds, while document terms are shown as blue circles; connecting lines are drawn in order to visualize the distances between projections of the matching index terms: *criteria*, *evaluation*, *retrieval* )

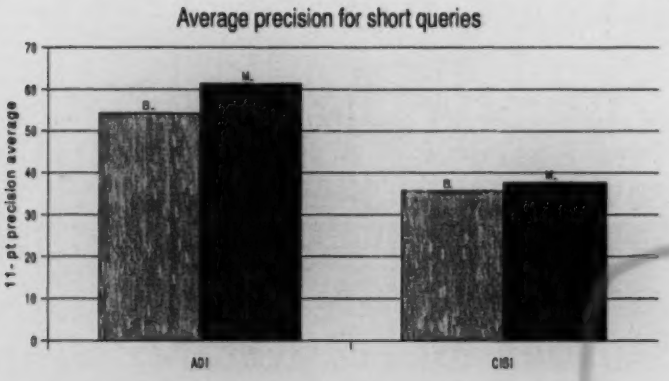



Figure 4.7: Performance of baseline (B.) and modified (M.) retrieval systems for the subset of short queries from ADI and CISI text collections

# Chapter 5

# Conclusions and future work

In this thesis we have described an approach for solving inexact graph matching problem using methods based on the projections of vertices, and their connections, into the eigenspaces of graphs - and associated clustering methods. Our analysis points to deficiencies of recent eigenspectra methods, such as the lack of uniqueness in purely spectral approaches and difficulties in dealing with matching of graphs of substantially different sizes, though demonstrates just how powerful full eigenspace methods can be for providing an alternative solution or serving as filters for such computationally intense (NP-complete for subgraph isomorphism matching) problems.

The proposed eigenspace projection clustering (EPC) method for graph matching relies on the adjacency matrix representation of graphs and deploys computationally efficient mathematical procedure of eigendecomposition, which allows it to be practically useful for large scale problems. Moreover, since the eigenspace projection technique normally requires only a small number of the most significant eigenvalues and their corresponding eigenvectors to be calculated, the procedure can be further optimized by taking into account this property. In comparison to the previously developed solutions from the spectral domain, the proposed EPC method has an important advatage of being able to match graphs with substantially different number of vertices, which obtains as a result of using two stages of normalization operations. Furthermore, a new correspondence clustering algorithm that is designed to be an integral part of the matching procedure enables the EPC method to discover a richer set of correspondence relationships, such as associations among structurally similar groups of vertices and subgraphs, in addition to the one-to-one type of vertex correspondence. It should also be mentioned that the developed clustering algorithm itself can be used

separately from the EPC method for solving the problem of clustering objects from two distinct datasets, between subsets of which a correspondence relation needs to be established.

The formulation of the EPC method is remarkably general, which allows it to be adapted for a variety of practical applications that involve matching graphs and generic relational structures that can be represented in the form of an adjacency matrix of a weighted graph. In the thesis presented, this property of the method is demonstrated by applying it to two different problem settings that originate from the domains of image and text processing. In the first application, the shock graph representations of various two-dimensional shapes were matched with EPC method. The obtained results compare favourably with those of the previously developed solutions. In addition to the numeric measure of shape similarity required for ranking different images, the proposed EPC method was shown to be able to provide an extra benefit of establishing meaningful correspondence relationships between the parts of the shapes being matched (as represented by subgraphs and groups of vertices in their respective shock graphs). In the second application, the problem of textual information retrieval was considered. The developed EPC method (or, more precisely, "EP" instead of "EPC", since the clustering facility of the method was not needed in this application) was used to incorporate the information about syntactic relations among keywords into the process of matching documents and queries in an attempt to improve retrieval results. For the modified IR system, the experiments showed a non-uniform performance and a marginal overall improvement. However, while lagging behind on mistakenly parsed, ungrammatical and complex/large queries, the system demonstrated a consistently better performance for a subclass of short queries, i.e. the cases where syntactic structure of the relations among keywords was indeed important. All things considered, it can be concluded that the encouraging results achieved in both of the applications confirm the properties of the EPC method of being general, easily adaptible and practically useful.

As for the extensions and future work, many areas obviously require improvement. First of all, even though the adjacency matrix representation format allows for various families of weighted graphs to be used with the EPC method, the procedure still remains a purely structural one. That is, an appropriate framework for working in a uniform fashion with both vertex and attributes, as opposed to dealing only with edge weights, has yet to be developed. For the shape matching application, an additional research effort would be

needed in designing a common matching index that could be appropriate for working with large shape databases. In the presented method implementation indexing is problematic, since the matching operation is only pairwise hence not scalable for very large collections. For the text-based IR applications, it may be considered to refine the method for a particular class of queries shown to improve the retrieval results, and apply it selectively. And, of course, the general nature of the proposed method's formulation, suggests that various other application areas should be explored, such as those in the fields of question-answering systems, natural language interfaces to relational databases, structural image partitioning and segmentation, handwritten character recognition, and others.

# Bibliography


[1] M. Berry, S. Dumais, and G. O'Brien. Using linear algebra for intelligent information retrieval. Technical Report UT-CS-94-270, 1994.

[2] B. Bhanu and J. Ming. *TRIPLE: A multi-strategy machine learning approach to target recognition.* Morgan Kaufmann Publishers, 1988.

[3] H. Bunke. Recent advances in structural pattern recognition with application to visual form analysis. *IWVF4, LNCS*, 2059:11–23, 2001.

[4] L. Collatz and U. Sinogowitz. Spektren endlicher grafen. *Abh. Math. Sem. Univ. Hamburg*, 21:63–77, 1957.

[5] L. Cordella, P. Foggia, C. Sansone, and M. Vento. Performance evaluation of the vf graph matching algorithm. *Proc. of the 10th ICIAP*, pages 1172–1177, 1999.

[6] D. Corneil and C. Gotlieb. An efficient algorithm for graph isomorphism. *Journal of the ACM*, 17:51–64, 1970.

[7] A. Cross, R. Wilson, and E. Hancock. Inexact graph matching using genetic search. *Pattern Recognition*, 30(6), 1997.

[8] P. Dimitrov, C. Phillips, and K. Siddiqi. Robust and efficient skeletal graphs. *Conference on Computer Vision and Pattern Recognition*, june 2000.

[9] M. Eshera and K. Fu. A graph distance measure for image analysis. *IEEE Transactions on Systems Man and Cybernetics*, 14(3), 1984.

[10] G. Fasano and Franceschini. A multidimensional version of the Kolmogorov-Smirnov test. *Monthly Not. R. Astron. Soc.*, (225):155–170, 1987.

[11] J. Feng, M. Laumy, and M. Dhome. Inexact matching using neural networks. *Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid Systems*, 1994.

[12] M. Garey and D. Johnson. *Computers and Intractability.* W. H. Freeman and Co., 1979.

[13] X. Jiang, A. Munger, and H. Bunke. On median graphs: properties, algorithms, and applications. *IEEE Trans. PAMI*, 23(10):1144–1151, October 2001.

[14] J. Kittler, W. Christmas, and M. Petrou. Probabilistic relaxation for matching of symbolic structures. *In H. Bunke, editor, Advances in Structural and Syntactic Pattern Recognition*, 1992.

[15] S. Lee and J. Kim. Attributed strokegraph matching for seal imprint verification. *Pattern Recognition Letters*, 9:137–145, 1989.

[16] D. lin. Dependency-based evaluation of minipar. *In Proceedings of the Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation*, 1998.

[17] B. Luo and E. Hancock. Structural graph matching using the em algorithm and singular value decomposition. *IEEE Trans. PAMI*, 23(10):1120–1136, October 2001.

[18] N. Maloy. Successor variety stemming: variations on a theme. 2000. project report (unpublished).

[19] Z. Marx, I. Dagan, and J. Buhmann. Coupled clustering: a method for detecting structural correspondence. *In: C. E. Brodley and A. P. Danyluk (eds.), Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, 2001.

[20] Z. Marx, I. Dagan, and E. Shamir. Detecting sub-topic correspondence through bipartite term clustering, 1999.

[21] B. McKay. nauty user's guide (version 1.5). Technical Report TR-CS90 -02, Computer Science Department, Australian National University, 1990.

[22] B. Messmer. *Efficient Graph Matching Algorithms for Preprocessed Model Graphs.* PhD thesis, University of Bern, CH, Institute for Applied Mathematics, November 1995.

[23] B. Messmer and H. Bunke. Subgraph isomorphism in polynomial time. Technical Report IAM 95-003, 1995.

[24] M. Mittendorfer and W. Winiwarter. Experiments with the use of syntactic analysis in information retrieval. *LNI, Proceedings, Series of the German Informatics Society (GI), Applications of Natural Language to Information Systems 6th International Workshop NLDB'01*, P-3, 2001.

[25] A. Myers and E. Hancock. Least commitment graph matching with genetic algorithms. *Pattern Recognition*, 34, 2001.

[26] M. Pelillo, K. Siddiqi, and S. Zucker. Matching hierarchical structures using association graphs. *IEEE Trans. PAMI*, 21(11), November 1999.

[27] D. Rouvray and A. Balaban. *Chemical applications of graph theory.* Academic Press, 1979.

[28] A. Sanfeliu and K. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 13(3), 1983.

[29] A. Schwenk. *Almost all trees are cospectral.* Academic Press, New York - London, 1973.

[30] G. Scott and H. Longuet-Higgins. An algorithm for associating the features of two patterns. *In Proc. Royal Society of London*, B244, 1991.

[31] L. Shapiro and J. Brady. Feature-based correspondence - an eigenvector approach. *IVC*, 10, 1992.

[32] A. Shokoufandeh and S. Dickinson. A unified framework for indexing matching hierarchical shape structures. *IWVF4, LNCS*, 2059:67–84, 2001.

[33] K. Siddiqi, S. Bouix, A. Tannebaum, and S. Zucker. Hamilton-jacobi skeletons. *To appear in International Journal of Computer Vision.*

[34] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 30:1–24, 1999.

[35] A. Smeaton. Using NLP or NLP resources for information retrieval tasks. In Tomek Strzalkowski, editor, *Natural language information retrieval*, pages 99–111. Kluwer Academic Publishers, Dordrecht, NL, 1999.

[36] S. Tirthapura, D. Sharvit, P. Klein, and B. Kimia. Indexing based on edit-distance matching of shape graphs. *Multimedia Storage and Archiving Systems III*, 3527(2):25–36, 1998.

[37] J. Ullman. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1), 1976.

[38] S. Umeyama. An eigen decomposition approach to weighted graph matching problems. *IEEE Trans. PAMI*, 10:695–703, 1998.

[39] URL. Adi test collection.
*ftp://ftp.cs.cornell.edu/pub/smart/adi.*

[40] URL. Cisi test collection.
*ftp://ftp.cs.cornell.edu/pub/smart/cisi.*

[41] C. J. vanRijsbergen. *Information Retrieval.* 2nd ed., Butterworths, 1979.

[42] E. Voorhees. Natural language processing and information retrieval. In *SCIE*, pages 32–48, 1999.

[43] S. Wu, Y. Ren, and C. Suen. Hierarchical atributed graph representation and recognition of handwritten chineese characters. *Pattern Recognition*, 24:617–632, 1991.

[44] J. Xu and W. Croft. Corpus-based stemming using co-occurence of word variants. *ACM Transactions on Information Systems*, 16(1):61–81, January 1998.

[45] H. Zha, X. He, C. Ding, M. Gu, and H. Simon. Bipartite graph partitioning and data clustering. *Proc. ACM 10th Int'l Conf. Information and Knowledge Management (CIKM 2001)*, 2001.

[46] W. Zwick and W. Velicer. Comparison of five rules for determining the number of components to retain. *Psychological Bulletin*, 99, 1986.

# Appendix A

# Aggregate relational lattice construction example

For the sake of this example, we will consider document 45 from ADI collection that consists of three parsable units: one title, and two sentences of text, as shown below.

**ADI collection, Document 45**

```
Title:
  Graduate training in information science : definitions and
  developments at the georgia institute of technology .
Text:
  the graduate degree program in information science at georgia tech
  is described .  areas of specialization within the curriculum and
  definitions of terms are given .
```

After being processed by the *Minipar* parser, all of the syntactic relations among the words in the document are output as pairs in the format given in the example below.

**Minipar output example**

```
...
area              N:mod:PREP       of
of                PREP:pcomp-n:N   specialization
specialization    N:mod:PREP       within
within            PREP:pcomp-n:N   curriculum
...
```

The above *Minipar* output excerpt shows various examples of "noun-preposition" type of syntactic relationships found by the parser. After the obtained stopwords are replaced by their part of speech (POS) tags, all of these pairs of relationships are then used to fill the common syntactic relational matrix. The matrix is symmetric and has as many rows and columns as there are unique keywords and POS tags in the whole document. For our example this list will consist of the following 19 keywords and POS tags: { *graduate, training, PREP, N, information, definition, development, DET, georgia, technology, C, describe, degree, BE, give, area, specialization, curriculum, term*}, and thus the matrix (which we refer to as "aggregate relational lattice" in the earlier dicussion) will have the size of 19 by 19 elements. When all of the relational data is combined together, it looks as follows.

**The aggregate syntactic realtional lattice for document 45**

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | 0 | 0.6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1.0 | 0 | 0.6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0.6 | 0 | 1.4 | 0 | 0.6 | 0.6 | 0 | 0 | 0.6 | 0 | 0 | 0 | 0 | 0 | 0.6 | 1.2 | 0.6 | 0.6 |
| 0.6 | 0 | 1.4 | 0 | 1.2 | 0 | 0 | 0.4 | 1.2 | 0 | 0 | 1.2 | 0.6 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0.6 | 0 | 0 | 0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0 | 0 | 0 | 0 |
| 0 | 0 | 0.6 | 0 | 0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0.4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.6 | 0 |
| 0 | 0 | 0 | 1.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0.6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.6 | 0 | 0 | 0.6 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0.6 | 0 | 0 | 0.6 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0.6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.6 | 0 | 0 | 0.6 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.6 | 0 | 0 | 0.6 | 0 | 2.0 | 0 | 0 | 0 |
| 0 | 0 | 0.6 | 0 | 0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2.0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0.6 | 0 | 0 | 0 | 0 | 0.6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0.6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The shown above matrix of relations is now in the final form as it is used by the information retrieval system to judge the degree of syntactic relevance between matching keywords in various queries and document 45 by applying the EPC method. The assignment strategy for its individual weights differentiates among syntactic relationships of various types, and sets these values as specified below.

$$weight(word_1, word_2) = \begin{cases} \mu_1 & \text{if } word_1 \text{ and } word_2 \text{ are stopwords/POS tags} \\ \mu_2 & \text{if either } word_1 \text{ or } word_2 \text{ is a stopword/POS tag} \\ \mu_3 & \text{if neither } word_1 \text{ nor } word_2 \text{ are stopwords/POS tags} \\ & \text{such that } \mu_1 < \mu_2 < \mu_3 \end{cases}$$

In the shown example the relation weights are set as follows: $\mu_1 = 0.2, \mu_2 = 0.6, \mu_3 = 1.0$